Facultad de Ingeniería
Instituto de Computación

Proyecto de Grado

Migración de SGS a plataforma "Open/Free"

por

Alexis Hernández,
Leonardo Méndez,
Martín Prunell.

Supervisores:	**Responsable:**	**Tribunal:**
Lorena Etcheverry,	Nelson Calero	Carlos Luna,
Ariel Sabiguero.		Adriana Marotta,
		Guillermo Monccechi

Montevideo, 2012

Resumen

Un sistema central, crítico y de extensa utilización en la Administración de Servicios de Salud del Estado (ASSE) es el Sistema de Gestión de Salud (SGS), actualmente denominado Atención Primaria (AP), con más de 200 puestos distribuidos en todo el país. Es una aplicación web desarrollada con Genexus utilizada diariamente por cerca de 300 funcionarios y con el potencial de gestionar servicios de salud a más de un millón de usuarios.

El backend actual de la aplicación es Oracle, tanto para la base transaccional como la de reporting. Siguiendo la decisión adoptada por ASSE de incorporar software libre, se quiere evaluar la madurez de los Sistemas de Gestión de Bases de Datos Relacionales (RDBMS por sus siglas en inglés *Relational Database Management System*) libres y de código abierto (FOSS por sus siglas en inglés *Free and Open Source Software*) para dar soporte a un sistema de gran porte como el SGS dentro del contexto del Estado uruguayo.

El objetivo es poner el SGS en funcionamiento sobre una o más de las soluciones FOSS disponibles en el mercado migrando la base de datos (datos, esquema de tablas, triggers y stored procedures). Mediante este experimento y el de la ejecución de algunos procedimientos relevantes para ASSE sobre el sistema, a saber: respaldos en caliente, optimización de consultas, administración y monitoreo, se espera generar información valiosa como insumo para la toma de decisiones a la hora de evaluar soluciones para realidades que presenten similitudes con esta.

El presente trabajo responde, en base a un análisis técnico de soluciones vigentes y a una migración completa de un aplicativo crítico la pregunta de si se puede instalar, operar y soportar un sistema de la escala de AP sobre RDBMS del tipo FOSS.

Palabras clave: Bases de Datos, Open Source, Free Software, Genexus.

Índice general

Capítulo 1

Introducción

1.1. Contexto

La Administración de los Servicios de Salud del Estado (ASSE) se encuentra en un proceso de informatización, habiendo incorporando el uso de software para dar apoyo a la gestión de los servicios de salud en Montevideo y Área Metropolitana y apuntando a incorporarlo también en el resto del país.

ASSE cuenta actualmente con varios aplicativos para la gestión y administración de los diversos servicios. Este trabajo se enfoca en el Sistema de Gestión de Salud (SGS), actualmente denominado Atención Primaria.

El Sistema de Gestión de Salud

El Sistema de Gestión de Salud (SGS) es una aplicación web desarrollada inicialmente para el apoyo en la atención de Red de Asistencia Primaria, extendiéndose paulatinamente con el tiempo. Hoy en día permite la gestión de emergencias, policlínicas y farmacias, así como la creación de agendas para los médicos y la asignación de números a los pacientes.

La base de datos del SGS contiene alrededor de 2500 usuarios registrados para utilizar el sistema informático, entre ellos personal médico y administrativo. Aquí el término "usuario" del sistema debe distinguirse de "beneficiario" del sistema, término que refiere a los pacientes. Se calcula que existen aproximadamente 300 conexiones activas a la base de datos en un instante dado. Estas características lo convierten en un sistema estatal de gran porte. Se prevé que el SGS continúe expandiéndose y sea utilizado cada vez en más centros de salud en todo el país.

El sistema está desarrollado con la herramienta Genexus, generando código Java y utiliza la versión 10g de Oracle como *back end*. La aplicación corre en un servidor Tomcat y los usuarios están obligados a utilizar Internet Explorer como navegador web debido a un requerimiento de localización, véase el Anexo A.

La base de datos tenía un tamaño cercano a los 60 GB al comenzar el proyecto, habiendo crecido aproximadamente 20GB en el último año. Algunas de sus tablas del esquema tienen una cantidad de registros del orden de varias decenas de millones, y la aplicación hace un uso intenso de *stored procedures*. Por otro lado, existe un gran número de *triggers* utilizados para registrar datos de auditoría.

Arquitectura

El SGS funciona sobre cuatro instancias de Oracle, cada una de ellas con una función específica:

- Oracle 1: Instancia principal en la que se encuentra la base de datos con la que interactúa el sistema SGS.
- Oracle 2: Base de datos de contingencia con actualización.
- Oracle 3: Respaldo diario de Oracle 1. Sirve la base de datos al sistema gerencial.
- Oracle 4: Histórico. Sirve la base de datos al sistema histórico

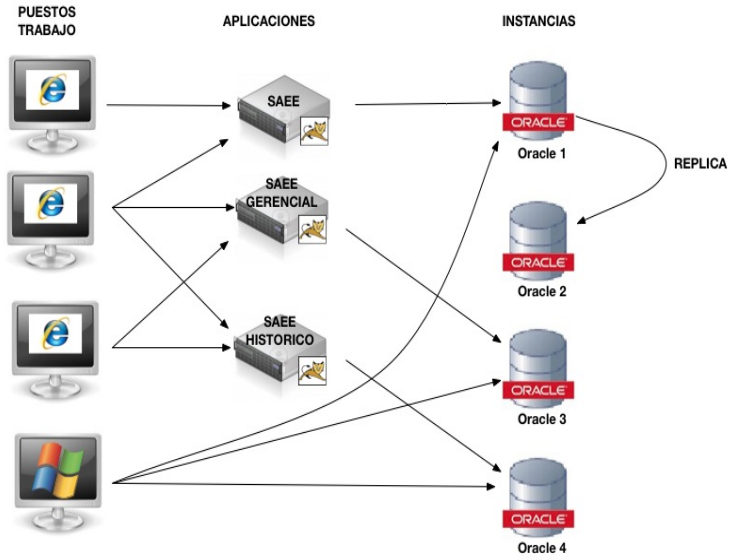

Figura 1.1: Arquitectura

La figura 1.1 muestra un esquema de la arquitectura actual, las cuatro instancias funcionan de la siguiente manera: En primer lugar, se busca que el trabajo *online* se realice sobre la instancia principal dado que allí se encuentra la base de datos operativa y actualizada.

La segunda instancia cumple con una función de respaldo tipo espejo. Mantiene la misma información que la instancia principal pero no es una base de datos que sea accesible en el uso cotidiano. Su función principal es la de respuesta a contingencias con la base de datos de la instancia principal.

La tercera instancia es un respaldo diario. Sirve a los reportes gerenciales que no necesitan la información *online*, quitándole carga a la instancia principal y distribuyéndola hacia una nueva instancia. Por último, la información histórica, es derivada hacia una cuarta instancia que se encarga de reducir el volumen de datos de la instancia principal. Esta cuarta instancia permite mantener los datos históricos accesibles.

1.2. Motivación

El software FOSS está adquiriendo una robustez y madurez suficiente como para competir con cualquier software privativo. Esto se debe principalmente a la libre disponibilidad del código y a toda la información (estado, errores, etc.) subyacente. La gran comunidad de usuarios y desarrolladores hacen una importante diferencia, debido a la diversidad de usuarios los productos están usualmente muy bien probados y se puede obtener ayuda y consejo de forma rápida.

Gigantes de la industria tales como Facebook [I] utilizan Sistemas de Bases de Datos de este tipo. Sin embargo, esto no es suficiente para asegurar el éxito en el contexto ASSE debido a diferencias que pueden existir entre la cultura y posibilidades de una organización privada o extranjera y la del Estado uruguayo.

En el año 2008, el directorio de ASSE tomó la decisión de promover la utilización de software libre en el organismo. De acuerdo a este lineamiento, se desea evaluar la utilización de Sistemas de Bases de Datos Relacionales (RDBMS) FOSS como *back end* de las aplicaciones a futuro. La utilización de software libre tiene como atractivo adicional la ausencia de costo económico por licencias, proveyendo de una mayor flexibilidad para elegir la cantidad de instancias que se necesiten a la hora de planificar, desarrollar y liberar un sistema.

1.3. Objetivos

En esta realidad, el objetivo general de este proyecto es generar información valiosa para dar apoyo a la toma de decisiones y fundamentalmente evaluar la capacidad de utilizar un RDBMS FOSS como *back end* para dar soporte al SGS. Dichos insumos se obtienen del ejercicio de migrar el SGS a un RDBMS FOSS y de la ejecución de algunos procedimientos de administración y mantenimiento relevantes para ASSE.

No es la intención de este documento, generar información acerca de una migración genérica de una base de datos Oracle a un RDBMS FOSS, ni llegar a conclusiones acerca de una migración total del sistema de base de datos utilizado actualmente. Se buscará evaluar si los RDBMS analizados cumplen con las necesidades del SGS que son cubiertas al día de hoy por Oracle. Queda fuera del alcance de este proyecto cualquier análisis acerca de las cualidades de los sistemas de bases de datos con los que se trabajará que se abstraiga de la realidad concreta de la aplicación a migrar.

En resumen los dos ejes principales en los que se centra este trabajo son: por una parte, el análisis de las funcionalidades ofrecidas por cada uno de los sistemas de base de datos estudiados y, por otra parte, las posibilidades que puede ofrecer la utilización de Software Libre en el contexto de estudio.

Las tareas que surgen de la planificación del trabajo son:

1. Elegir los RDBMS FOSS candidatos
2. Relevar herramientas de administración y monitoreo
3. Verificar la factibilidad de realizar procedimientos de mantenimiento relevantes para ASSE

[I] http://www.infoq.com/presentations/Facebook-Software-Stack

4. Migrar esquemas, *stored procedures*, *triggers* y datos existentes

5. Evaluar el funcionamiento del SGS sobre los nuevos RDBMS.

1.4. Aportes

Los aportes de este proyecto deben evaluarse en dos sentidos diferentes. El más lineal consiste en evaluar el éxito o no de la migración desde la perspectiva del SGS.

Sin embargo, hay una segunda manera de mirar el proyecto, observando cuáles son los aprendizajes que se desprenden del ejercicio de la migración de la base de datos. Este nuevo sentido tiene una importancia mayor pues intenta ampliar los márgenes de aplicabilidad de los resultados. Al momento de la realización del proyecto, no existen indicios de que ASSE pretenda migrar efectivamente la base de datos, por lo que limitarse a evaluar el éxito o no según el funcionamiento de la aplicación, significaría evaluar un resultado poco significativo. Por tanto, se pretende que uno de los aportes del proyecto sea servir como insumo que, situándose en un contexto concreto - el del SGS - permita que los problemas, las dificultades y las soluciones encontradas durante el desarrollo del trabajo faciliten el desarrollo de otros proyectos que presenten analogías.

1.5. Organización del Documento

Tras este capítulo introductorio, el presente documento se organiza de la siguiente manera: en el segundo capítulo se expondrán los principales requerimientos planteados, así como los RDBMS elegidos para las pruebas de migración. Junto a éstos, se presentan algunas de las herramientas disponibles para administración y monitoreo de las bases de datos a analizar.

El tercer capítulo investiga y comprueba el funcionamiento de algunos procedimientos relevantes para el DBA de ASSE. Se abordan aquí algunos de los problemas encontrados, así como las diferencias fundamentales encontradas entre los distintos DBMS analizados.

El cuarto capítulo presenta el proceso concreto de migración. Entre las problemáticas principales a resolver se destacan:

- la creación de un esquema completo de la base de datos para cada uno de los DBMS libres elegidos

- la realización de una migración completa entre Oracle y cada uno de dichos DBMS

- la minimización del tiempo que la base de datos deberá estar *offline* durante la migración

Además, se detallan las dificultades concretas que se enfrentaron durante el proceso de migración de los datos, el proceso de migración de los *stored procedures* utilizados por la aplicación y los ajustes a ésta que se realizaron para su adecuado funcionamiento.

El capítulo quinto resume los resultados de las pruebas realizadas, el sexto plantea las conclusiones del trabajo y el séptimo capítulo, las líneas de investigación que podrían seguirse a futuro.

1.5.1. Público Objetivo

Este documento está orientado a profesionales de informática encargados de la toma de decisiones. Desarrolla los temas abarcados con la profundidad necesaria para dar una idea clara y concisa de las posibilidades de utilización de FOSS. Adicionalmente, se describen detalles técnicos de soluciones a problemas encontrados durante el desarrollo del proyecto que resultarán muy interesantes al profesional especializado en esta área, tales como técnicos informáticos, desarrolladores Genexus y personal de soporte de Bases de Datos.

Capítulo 2

Elección de Bases de Datos y Resumen de Herramientas Disponibles

2.1. Introducción

Habiendo visto el contexto en el cual se desarrolla el proyecto y los objetivos que se buscan alcanzar, se releva aquí la situación actual de los RDBMS a utilizar. Esto implica, por una parte, la evaluación teórica de las funcionalidades disponibles en cada uno de los manejadores en relación con el SGS y, en segundo lugar, el relevamiento de las herramientas de administración y monitoreo disponibles para cada uno de ellos.

No es objetivo de este capítulo realizar una selección de herramientas ni una revisión comparativa. La realización de este relevamiento, parte de los requerimientos planteados por el SGS al motor actual de base de datos. Tras esto, se contrasta la lista de los requerimientos con las funcionalidades disponibles en cada uno de los RDBMS escogidos, evaluando en cada uno de los casos si dichas funcionalidades cubren las necesidades planteadas por el sistema actual.

2.2. Principales requerimientos

Para el estudio de factibilidad de la utilización de RDBMS FOSS se han de tener en cuenta los siguientes requerimientos:

- soporte completo de las características ACID
- implementación de *stored procedures* y *triggers*.
- disponibilidad de la base de datos frente a fallas de software y/o hardware
- integración con Genexus.

2.3. RDBMS Evaluados

En las últimas décadas las herramientas FOSS han logrado una mayor difusión y aceptación a nivel internacional debido, principalmente, al constante desarrollo de comunidades de programadores y empresas.

Los manejadores de base de datos escogidos para la evaluación forman parte de los sistemas de bases de datos relacionales RDBMS y pueden clasificarse en dos grupos. El primero está constituido por los sistemas tradicionales o de almacenamiento por fila, mientras que el segundo representa al paradigma de los sistemas columnares.

2.3.1. RDBMS tradicionales

Los RDBMS tradicionales implementan el modelo relacional y están orientados a dar el mejor rendimiento posible en sistemas del tipo de transacciones *online* (OLTP). El almacenamiento de los datos se hace en forma secuencial, de manera que cada registro almacenado se sitúa a continuación del anterior.

Considerando que el objetivo del presente proyecto es evaluar la capacidad de los RDBMS libres de dar soporte a un sistema de gran porte, se decidió evaluar -dentro de este grupo- los dos RDBMS FOSS que cuentan con soporte por parte de Genexus: MySQL y PostgreSQL.

2.3.1.1. MySQL

MySQL es un sistema de almacenamiento multimotor con una conocida historia el el desarrollo de aplicaciones web. Se ofrece bajo licencia GNU GPL para cualquier uso compatible con esta, pero existe la posibilidad que los interesados en recibir un soporte adicional y otras herramientas puedan adquirirlo en una licencia específica para este uso [4].

En este proyecto, la versión utilizada es la 5.1 debido a que se considera que tiene el tiempo de maduración suficiente en el mercado.

Motores de Almacenamiento ("Storage Engines") evaluados

Una de las características más destacada que presenta MySQL es la posibilidad de utilizar distintos motores de almacenamiento. A través de una misma interfaz permite la utilización de diferentes implementaciones en la forma de tratar los datos, respetando el paradigma de base de datos relacional [11].
Cada motor de almacenamiento posee cualidades específicas ya sea en tiempos de recuperación de datos, seguridad, interacción con otras bases de datos, permitiendo un mejor aprovechamiento de los recursos.

Los motores se pueden definir a nivel de tablas; es decir que una misma base de datos puede operar sobre varios motores de almacenamiento en paralelo. A continuación se detallan los principales motores de MySQL.

MyISAM - Es el motor de almacenamiento por defecto. Se basa en el código ISAM pero tiene muchas extensiones útiles. Se caracteriza principalmente por ofrecer una lectura y escritura rápidas. Además, es el único motor que posibilita la búsqueda *fulltext*, es decir, permite buscar subcadenas de texto sobre un campo. La carencia más importante de MyISAM es que no incluye soporte para transacciones.

InnoDB - Proporciona tablas transaccionales (ACID *compliant*) para MySQL con operaciones de *commit*, *rollback*, y capacidad de recuperación para proteger

los datos de usuario. Provee bloqueo a nivel de fila y lecturas sin bloqueos que aumentan la concurrencia multiusuario y el rendimiento. InnoDB almacena los datos de usuario en índices clusterizados para reducir la E/S de consultas comunes basadas en claves primarias. Para mantener la integridad de los datos, InnoDB soporta claves foráneas y restricciones de integridad referencial.

NDBCluster - Es el motor de almacenamiento usado por MySQL Cluster[20] para implementar tablas que se particionan en varias máquinas. Es particularmente adecuado para aplicaciones que requieren el más alto grado posible de tiempo online y de disponibilidad. Al no manejar claves foráneas, requiere que las restricciones planteadas por éstas se trasladen a *triggers* desarrollados ad hoc.

Archive - Es un motor orientado al manejo de datos históricos. Dado que se presume una baja tasa de accesos a la información almacenada en estas tablas, el motor Archive no maneja índices y los archivos de datos son comprimidos.

Memory - También llamado *Heap*, reside en memoria y utiliza índices de tipo *hash*. Esta particularidad le permite ser un motor extremadamente rápido, pero cuando se detiene el servicio se pierden los datos contenidos en las tablas.

El cuadro 2.1 muestra una comparativa entre las funcionalidades de los distintos motores:

Motor	MyISAM	Memory	InnoDB	Archive	NDBCLUSTER
Límites de almacenamiento	256TB	RAM	64TB	None	384EB
Transacciones	No	No	Sí	No	Sí
Granularidad de locks	Table	Table	Row	Row	Row
MVCC	No	No	Sí	No	No
Soporte tipo de datos geoespacial	Sí	No	Sí	Sí	Sí
Soporte de índices geospaciales	Sí	No	No	No	No
B-tree	Sí	Sí	Sí	No	Sí
Índices hash	No	Sí	No	No	Sí
Índices de búsqueda "Full-text"	Sí	No	No	No	No
Índices clusterizados	No	No	Sí	No	No
Caché de datos	No	N/A	Sí	No	Sí
Caché de índices	Sí	N/A	Sí	No	Sí
Datos comprimidos	Sí [a]	No	Sí[b]	Sí	No
Datos encriptados[c]	Sí	Sí	Sí	Sí	Sí
Soporte de clusters	No	No	No	No	Sí
Soporte de replicación[d]	Sí	Sí	Sí	Sí	Sí
Soporte de claves foráneas	No	No	Sí	No	No
Respaldo / Recuperación PIT[e]	Sí	Sí	Sí	Sí	Sí
Query cache support	Sí	Sí	Sí	Sí	Sí
Actualización de estadísticas para el diccionario de datos	Sí	Sí	Sí	Sí	Sí

[a]La compresión en las tablas MyISAM está soportada solo cuando se usa el formato de compresión de fila. Dichas tablas son de sólo lectura.

[b]Las tablas comprimidas en InnoDB solo están soportadas por el plugin InnoDB.

[c]Implementada en el servidor (vía funciones de encriptación) más que en el motor de almacenamiento.

[d]Implementada en el servidor más que en el motor de almacenamiento.

[e]Implementada en el servidor más que en el motor de almacenamiento. PIT ("Point in time recovery"): Recuperación de los datos en un momento determinado.

Cuadro 2.1: Tabla comparativa de los Motores de Almacenamiento en MySQL

Para este proyecto la principal cualidad que se espera de los motores de base de datos es que soporten la propiedad de ser transaccionales, por lo que, en una primera selección,

los dos motores candidatos son InnoDB y NDBCluster.

Si bien los dos motores preseleccionados cumplen con las propiedades ACID, se debió descartar al motor NDBCluster debido a que está orientado a correr en una base de datos distribuida en varios servidores, agregando funcionalidades degradatorias del rendimiento e innecesarias en el caso de prueba de este proyecto[1].

2.3.1.2. PostgreSQL

PostgreSQL fue concebido en el ámbito académico y luego lanzado al mundo FOSS, recibiendo la contribución de una importante comunidad de desarrolladores. Rápidamente ganó fama por su gran estabilidad y robustez, logrando el apoyo de grandes corporaciones y sobre todo por organizaciones gubernamentales de los Estados Unidos[25]. En el contexto de este proyecto se utiliza la versión 8.4.5.

El cuadro 2.2 presenta un resumen de algunas de las características ofrecidas por el motor de PostgreSQL como comparación con las presentadas anteriormente para los motores de MySQL.[24]

Límite	Valor
Tamaño máximo de la base de datos	Ilimitado
Tamaño máximo de tabla	32 TB
Tamaño máximo de fila	1.6 TB
Tamaño máximo de campo	1 GB
Cantidad de filas por tabla	Ilimitada
Columnas por tabla	250 - 1600 dependiendo del tipo de columna
Cantidad de índices por tabla	Ilimitado

Cuadro 2.2: Características de PostgreSQL

Adicionalmente, PostgreSQL cumple con el estándar ANSI-SQL:2008. Tiene soporte completo para subconsultas (incluyendo *subselects* en la cláusula FROM), niveles de aislamiento READ COMMITTED (lectura confirmada) y SERIALIZED (serializado) . PostgreSQL tiene un catálogo de sistema completamente relacional que soporta múltiples esquemas por base de datos, el cual es accesible a través de la información de esquema como se define en el estándard de SQL.

Entre las características de control de integridad de datos se implementan claves primarias, claves foráneas con actualizaciones y eliminaciones restrictivas y en cascada, control de *constraints* de unicidad y de NOT NULL. Soporta además índices compuestos, únicos, parciales y funcionales que pueden utilizar cualquiera de los siguientes métodos de almacenamiento: B-tree, R-tree, hash y GIST[35].

Adicionalmente, PostgreSQL cuenta con soporte para un gran número de lenguajes de programación.

2.3.2. RDBMS con almacenamiento columnar

En los últimos años se puede observar un crecimiento importante en el desarrollo de bases de datos con características no tradicionales, siendo un ejemplo el de los sistemas

[1] http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster-limitations-performance.html

manejadores de bases de datos con almacenamiento columnar. Un sistema de bases de datos columnar almacena la información organizada en columnas en contraposición a filas como lo hacen los sistemas tradicionales.

Para implementaciones que hacen uso masivo de información -sistemas OLAP-, esta es una organización de datos mucho más eficiente. Mientras que el almacenamiento por filas es extremadamente eficiente para escrituras, ya que agregar una fila de datos a una tabla requiere simplemente agregar bloques a un archivo en disco, el almacenamiento columnar tiene un mejor rendimiento para consultas de lectura complejas.

Un ejemplo es el cálculo del promedio de valores de alguna columna. En un RDBMS tradicional, el sistema deberá leer todas las filas de la tabla y luego filtrar por la columna objetivo para poder realizar el cálculo. Si la tabla tiene muchas columnas, se realizan más operaciones a disco que las que podrían hacerse si se pudiera acceder directamente a los datos de la columna buscada. Esta optimización es la que implementan los motores almacenamiento columnar. En estos casos, el manejador trae de disco solamente la columna que contiene los valores para considerar, disminuyendo notoriamente la cantidad de accesos.

Adicionalmente, dado que los tipos de datos de una misma columna son uniformes, el almacenamiento columnar facilita la compresión de datos con respecto al almacenamiento por filas. Además podría evitarse el almacenamiento de los valores NULL. Un almacenamiento por filas no puede omitir columnas en operaciones de selección, y para poder proveer acceso aleatorio a los datos, estos deben tener un tamaño fijo, es decir, se debe reservar el mismo espacio en disco para todos los valores, incluidos los NULL. En el almacenamiento por columnas se guardan los valores reales utilizando el espacio exacto que estos ocupan, pudiendo omitir los valores NULL.

Como contrapartida, para insertar una fila en una tabla se requieren más operaciones en disco, ya que se deberá actualizar cada una de las columnas de la tabla, haciendo que este tipo de almacenamiento no sea adecuado para aplicaciones de escritura intensiva[38].

De una primera revisión de las implementaciones FOSS de motores columnares disponibles, se encontró que MonetDB era la que presentaba mayor documentación disponible. Por otra parte el manejo de transacciones lo diferencian de otros que, como LucidDB no las soportan[II] o lo hacen de manera no tradicional como C-Store[III]. Se acordó, por tanto, seleccionar a MonetDB para hacer un ensayo con motores columnares.

2.3.3. MonetDB

MonetDB es sistema de manejador de base de datos open source de alta performance desarrollado en el Instituto Nacional de Investigación para Matemáticas y Ciencias de la Computación (CWI; Centrum voor Wikunde en Informática) en Holanda. Se diseñó para proveer alta performance en consultas complejas contra bases de datos de gran tamaño.

La familia MonetDB consiste de:

- MonetDB/SQL: implementación del modelo relacional
- MonetDB/XQuery: implementación del modelo XML
- MonetDB Server: servidor de base de datos multimodelo.

[II] http://www.luciddb.org/wiki/LucidDbConcurrencyControl
[III] http://db.csail.mit.edu/projects/cstore/

La arquitectura de MonetDB hace énfasis en que múltiples *frontends* pueden conectarse con el *backend*. Se puede ejecutar consultas relacionales sobre el *frontend* y tener MonetDB Server como *backend*. El lenguaje intermedio entre las partes es el MIL (MonetDB Interpreter Language).

MonetDB utiliza un modelo de tablas binario donde todas las tablas consisten de exactamente dos columnas. Estas se llaman Tablas de Asociación Binarias BAT. Cada *frontend* utiliza reglas de correspondencia para asociar modelos de datos lógicos como se ven en el *frontend* con datos binarios en el *backend*.

MonetDB provee soporte para SQL, transacciones y una amplia gama de interfaces de programación. Sin embargo, Genexus no cuenta con un conector para este RDBMS. Este factor es determinante para descartar MonetDB en forma temprana. Los pormenores sobre el intento de integración con MonetDB serán discutidos en el Apéndice F.

2.4. Herramientas consideradas

El objetivo de esta sección es buscar el apoyo necesario a las tareas relevantes para ASSE entre las herramientas disponibles en la comunidad FOSS. No se trata de una evaluación comparativa de las herramientas disponibles, sino de una presentación de las características de aquellas que ofrecen un mejor abanico de funcionalidades.

El procedimiento seguido para la evaluación fue la instalación de cada una de las herramientas libres, así como la instalación de versiones de prueba de las ofrecidas con licenciamiento comercial.

Es de esperar que esta sección pueda ser útil al DBA que busca herramientas de administración o para quien deba decidir por un motor u otro según las herramientas de administración que encuentre disponibles.

2.4.1. Herramientas para MySQL

Se revisarán algunas de las herramientas que se consideraron más útiles e interesantes disponibles para MySQL, tanto para administración como para monitoreo.

2.4.1.1. Herramientas de Administración

MySQL distribuye en forma gratuita, bajo Licencia GPL, un conjunto de herramientas gráficas disponibles para descarga en su sitio web, éstas se denominan MySQL Query Browser, MySQL Administrator, MySQL Migration Toolkit y MySQL Workbench, siendo este último una recopilación de las anteriores.

MySQL Query Browser[IV] permite crear, ejecutar y optimizar consultas a la base de datos por medio de sus diversas funcionalidades y está disponible desde la versión MySQL 4.0. Algunos ejemplos de funcionalidades son la exportación de los resultados de una consulta a diferentes formatos (cvs, xml, xls, etc.) y trabajar con registros Blob, permitiendo visualizar el contenido de la celda ya sea texto o imagen.

MySQL Administrator[V] permite realizar tareas administrativas sobre un servidor MySQL tales como configuración de las opciones de inicio del servidor, inicio y detención,

[IV] http://dev.mysql.com/doc/query-browser/es/index.html
[V] http://dev.mysql.com/doc/administrator/en

visualización de catálogos, monitoreo de conexiones al servidor, administración de usuarios, generación de estadísticas de uso, visualización de logs del servidor, como se ilustra en la figura 2.1.

MySQL Migration Toolkit[VI] permite la migración de esquemas y datos de varios sistemas de base de datos relacionales a MySQL y solamente está disponible para el sistema operativo Windows. Permite la migración desde base de datos Oracle, Microsoft SQL Server, Microsoft Access y Sybase.

MySQL Workbench[VII] permite el modelado de diagramas entidad-relación para bases de datos, además permite sincronizar el modelo con la base real mediante ingeniería inversa. Los modelos realizados con esta herramienta son compatibles con modelos DBDesigner 4.

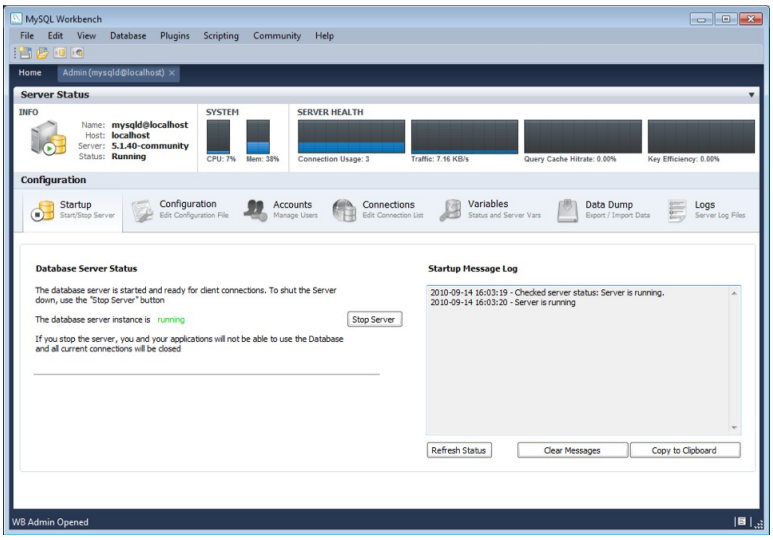

Figura 2.1: Captura de pantalla: Opciones de administración del servidor

A partir de la versión 5.2, MySQL Workbench ha integrado las características de editor de SQL e interfaz de administración de base datos englobando las funcionalidades provistas anteriormente por el conjunto de herramientas gráficas de MySQL.

Debido a que esta herramienta se presenta especialmente como evolución de MySQL Query Browser y de MySQL Administrator, se detallarán sus funcionalidades con más profundidad. En la figura siguiente se observa la variedad de opciones disponibles para la administración del servidor:

Desde esta ventana pueden realizarse la mayor parte de las tareas administrativas básicas, a saber:

- gestión de usuarios y privilegios
- configuración y parametrización del servidor

[VI] http://dev.mysql.com/doc/migration-toolkit/en
[VII] http://www.mysql.com/products/workbench

- monitoreo básico
- respaldos lógicos

En la figura 2.2, se muestra la facilidad ofrecida por MySQL Workbench para la edición de datos en las tablas de la base de datos:

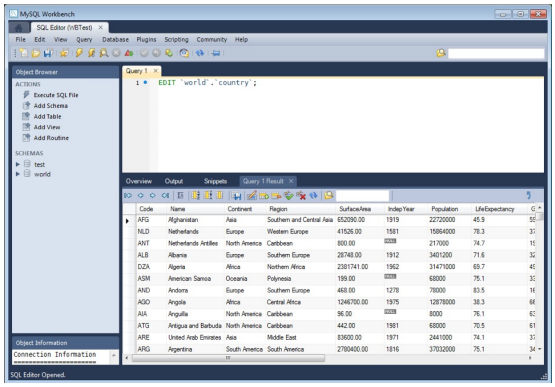

Figura 2.2: Captura de pantalla: Edición de tabla

Por último, la herramienta de modelado ofrecida permite tanto la generación de los scripts SQL necesarios para la creación de un esquema en base, como la operación de ingeniería inversa de modo de obtener el diagrama gráfico de una base del sistema (ver figura 2.3). Esto facilita de manera evidente el proceso de creación y modificación de las bases de datos, así como el análisis de las relaciones entre las tablas, las claves foráneas manejadas, índices, etc.

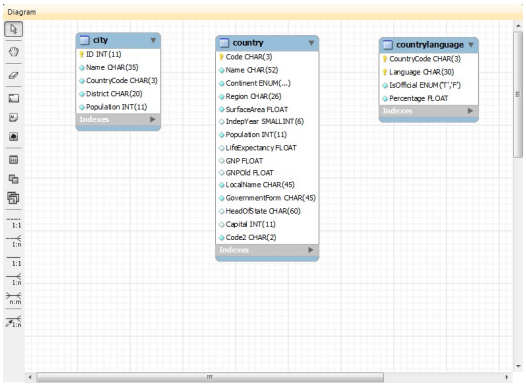

Figura 2.3: Captura de pantalla: Modelado

phpMyAdmin [VIII]es una herramienta web desarrollada en PHP, de libre distribución bajo licencia GPL. Es una herramienta que permite acceder a todas las funcionalidades típicas de la administración de una base de datos MySQL a través de una interfaz web. El hecho de que sea una aplicación web permite que sea multiplataforma, pudiendo

[VIII] http://www.phpmyadmin.net

ser accedida desde cualquiera de los navegadores web más populares, utilizando la autenticación de la base de datos para el acceso a los datos. Al momento de instalación y debido a su desarrollo en PHP, la aplicación consta de un conjunto de archivos los cuales pueden alojarse en un servidor web configurado con soporte para archivos PHP y mediante una simple configuración, acceder a los datos y estructuras.

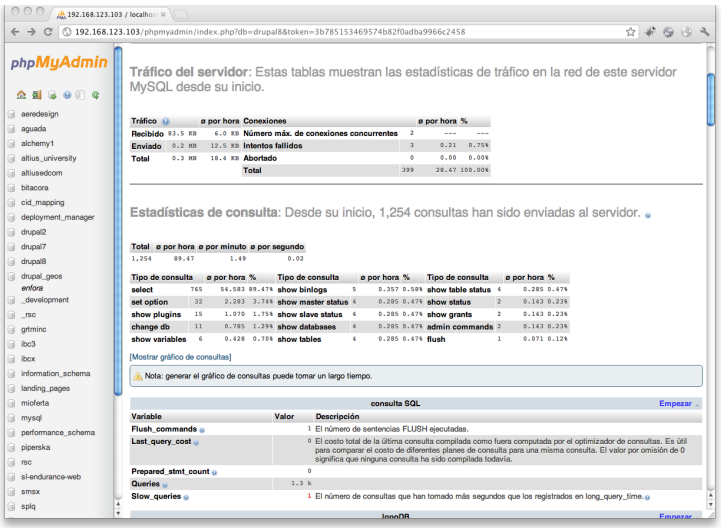

Figura 2.4: phpMyAdmin: Herramienta de Administración MySQL

Esta herramienta fue liberada por primera vez en setiembre de 1998 y a la fecha se encuentra en su versión 3.5, siendo liberada una nueva versión cada año esto gracias al aporte de una comunidad de usuarios muy grande, quienes desarrollan funcionalidades y liberan versiones en forma frecuente.

En la figura 2.4 se ilustra una captura de pantalla donde se observa las estadísticas globales de consultas que una base de datos MySQL ha recibido.

SQLyog [IX] es un software comercial específico para la administración de MySQL con diversas funcionalidades de acuerdo a tres diferentes ediciones, Final, Empresa y Profesional. El cuadro 2.3 muestra algunas de sus características:

[IX] http://www.webyog.com/en/

Características	Final	Empresa	Profesional
Redundantes Analizador Índice	sí	no	no
Esquema optimizador	sí	no	no
Analizador de consultas	sí	no	no
HTTP / HTTPS de túnel	sí	sí	sí
Túnel SSH	sí	sí	sí
Conexiones SSL	sí	sí	sí
SQL Formateador	sí	sí	no
Visual Query Builder	sí	sí	no
Esquema de diseño visual	sí	sí	no
Smart Autocompletar	sí	sí	sí
Finalización de código inteligente	sí	sí	si
Importar datos externos	sí	sí	no
Importar tareas se pueden programar	sí	sí	no
Esquema de sincronización	sí	sí	no
Comprimido Copias de seguridad programadas	sí	sí	no
Programador SQL y herramienta de informes	sí	sí	no
Trabajos programados	sí	sí	no

Cuadro 2.3: Características de SQLyog por edición

2.4.1.2. Herramientas de Monitoreo

Las herramientas de monitoreo permiten al administrador de base de datos, poder obtener una visión del estado de la base de datos. Mediante la presentación de un conjunto de datos, los cuales pueden ser mostrados de forma gráfica o en linea de texto se pueden detectar en forma rápida cualquier anomalía que puede perjudicar el rendimiento de la base de datos. A continuación se presentan las herramientas de monitoreo para MySQL más populares.

SHOW INNODB STATUS [x] es una sentencia SQL nativa de la base de datos MySQL que imprime una cantidad de indicadores internos del estado del motor de almacenamiento InnoDB. Con solo acceder a una consola SQL y mediante la ejecución de la sentencia, se obtiene una serie de datos que incluyen:

1. tablas y bloqueos de registros por cada transacción activa
2. esperas de transacciones
3. esperas de subprocesos
4. solicitudes de i/o
5. estadísticas del pool de *buffers*
6. actividades de descarga y mezcla del *buffer* en el proceso principal

[x] http://dev.mysql.com/doc/refman/5.0/es/innodb-monitor.html

INNOTOP [XI] fue escrita por Baron Schwartz en 2006 bajo licencia GNU en lenguaje Perl. Es una de las herramientas de monitoreo más útiles para los DBA de MySQL debido a que permite obtener información del estado de la base en tiempo real de un motor de MySQL InnoDB. Es posible contar con diferentes modos de monitoreo de acuerdo a lo que el DBA necesite:

1. **Estado de los *buffers*** muestra información sobre el estado del *Buffer Pool* ya este es uno de los *buffers* más importantes del motor de almacenamiento InnoDB, permitiendo contener los cache de datos e índices en memoria.

2. **Resumen de comandos** permite observar, datos estadísticos de los principales comandos ejecutados sobre la base de datos.

3. **Deadlocks** brinda información más detallada para poder diagnosticar por qué ocurrió un *deadlock* que fuera detectado por la base de datos (ver figura 2.5).

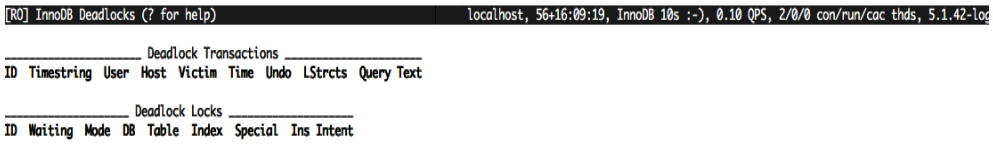

Figura 2.5: Captura de pantalla: Innotop - DeadLocks.

4. **Errores de claves foráneas**, brinda mayor información sobre los problemas padecidos con las claves foráneas.

5. **I/O**, cuando se sospecha de que existen problemas de rendimiento, esta opción puede dar información sobre lecturas y escrituras sobre el disco, tiempos de espera por una operación, tiempos de ejecución y estadísticas de los últimos accesos.

6. **bloqueos**, reporta qué tipos de bloqueos existen en el sistema, como así también cual fue el origen y con que frecuencia ocurre.

7. **Estado de replicación master/slave**, brinda información sobre el estado de la replicación si esta existe.

8. **Lista de consultas**, ilustrado en la figura 2.6 muestra información sobre las consultas que se encuentran ejecutando, también se puede obtener información particular a una consulta que allí se encuentre listada.

[XI] http://dev.mysql.com/doc/refman/5.1/en/innodb-tuning.html

Figura 2.6: Captura de pantalla: Innotop - Query List.

9. **Operaciones de filas y semáforos**, permite obtener estadísticas de filas insertadas, actualizadas, borradas y leídas así como la utilización de semáforos internos al motor de almacenamiento InnoDB.

10. **Transacciones**, esta vista brinda información sobre las transacciones que se encuentra en desarrollo, pudiendo observar que proceso es propietario, el tiempo que se encuentra en ejecución.

Además, InnoTop puede operar con múltiples servidores siendo útil para cualquier tipo de *clusters* de MySQL.

SHOW PROFILES [XII] es una utilidad que está disponible a partir de la versión 5 de MySQL e incorpora una funcionalidad que aparenta ser muy interesante para monitorear y detectar problemas de rendimiento en el servidor. Es la funcionalidad *Query Profiler*, la cual permite determinar cómo se distribuye el tiempo de ejecución de una consulta de acuerdo a cada una de sus etapas. Para poder utilizar este comando se debe establecer previamente la variable "profiling" a 1 como se ilustra en la figura 2.7

```
mysql> set profiling=1;
Query OK, 0 rows affected (0.00 sec)

mysql> select count(*) from usuario;
+----------+
| count(*) |
+----------+
|   330910 |
+----------+
1 row in set (0.50 sec)

mysql> show profiles;
+----------+------------+------------------------------+
| Query_ID | Duration   | Query                        |
+----------+------------+------------------------------+
|        1 | 0.49945400 | select count(*) from usuario |
+----------+------------+------------------------------+
1 row in set (0.00 sec)
```

[XII] http://dev.mysql.com/doc/refman/5.0/en/show-profiles.html

```
mysql> show profile for query 1;
+--------------------+----------+
| Status             | Duration |
+--------------------+----------+
| starting           | 0.000093 |
| Opening tables     | 0.000041 |
| System lock        | 0.000005 |
| Table lock         | 0.000009 |
| init               | 0.000014 |
| optimizing         | 0.000006 |
| statistics         | 0.000013 |
| preparing          | 0.000009 |
| executing          | 0.000007 |
| Sending data       | 0.498996 |
| end                | 0.000012 |
| query end          | 0.000002 |
| freeing items      | 0.000238 |
| logging slow query | 0.000005 |
| cleaning up        | 0.000004 |
+--------------------+----------+
15 rows in set (0.00 sec)

mysql> set profiling=0;
Query OK, 0 rows affected (0.00 sec)
```

Figura 2.7: Captura de pantalla: Comando Show Profiles.

CACTI [XIII] es un completo *frontend* RRDTool, que permite almacenar información necesaria para generar gráficas y poder monitorear el estado de un servicio en cualquier momento. Es una aplicación multiplataforma desarrollada en PHP bajo licencia GPL, que almacena la información en una base de datos MySQL. Cacti recopila los datos relevantes usando el protocolo SNMP de los servidores que se encuentren accesibles (ver figuras 2.8, 2.9 y 2.10. Para la generación de las gráficas, existen *templates* específicos para MySQL que permiten monitorear variables, estado de memoria, bloqueo de tablas, etc. Adicionalmente el DBA podrá programar *templates* personalizados apoyándose en una documentación muy completa[39].

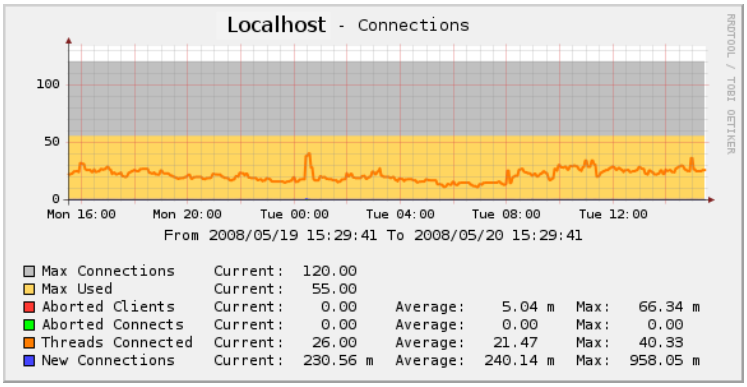

Figura 2.8: Cacti: conexiones

[XIII] http://code.google.com/p/mysql-cacti-templates/

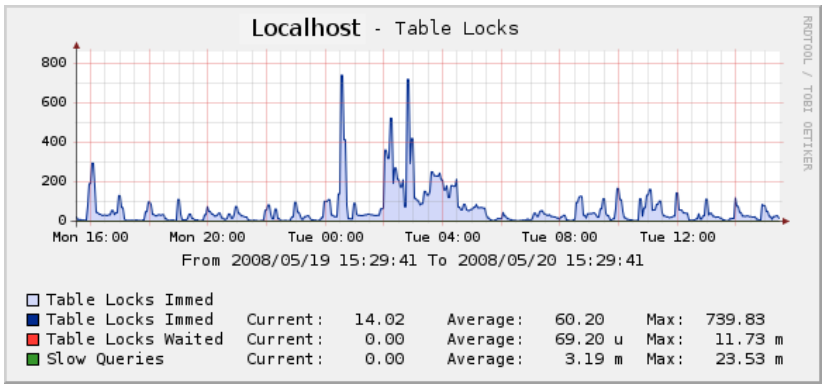

Figura 2.9: Cacti: locks de tablas

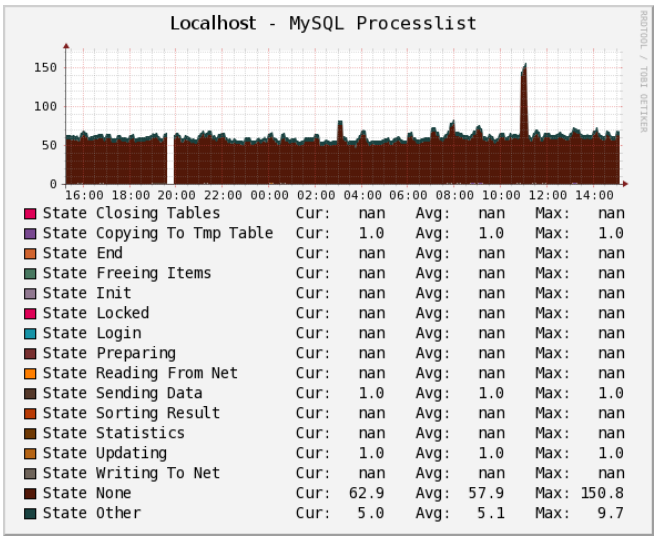

Figura 2.10: Cacti: lista de procesos

MySQL Enterprise Monitor [XIV] Esta herramienta que pertenece a la distribución Enterprise, ofrece un conjunto de funcionalidades que le serán útiles al DBA para evaluar resultados e implementar correcciones para lograr un mejor desempeño del motor. Permite el monitoreo en forma centralizada de múltiples bases de datos mediante la instalación de un agente en el servidor. Permite el análisis de consultas críticas de forma muy sencilla. Mediante la recopilación de métricas del estado del motor, Mysql Enterprise Monitor brinda advertencias sobre posibles problemas que puedan surgir, referidos a la memoria dinámica, cache, *hit ratios* y propone ajustes de las variables de configuración. El sistema de notificación, permite recibir las alertas tanto por el protocolo SNMP como por email permitiendo ser completamente configurable.

[XIV] http://www.mysql.com/products/enterprise/monitor.html

Percona Toolkit [XV] es un conjunto de herramientas de línea de comando avanzadas utilizadas por el equipo de soporte de Percona para realizar una variedad de tareas de MySQL y de sistema demasiado complejas para ser ejecutadas manualmente, entre las que se incluyen:

- Verificar consistencia de datos
- Archivar filas en forma eficiente
- Encontrar índices duplicados
- Resumir información sobre servidores de MySQL
- Analizar consultas a partir de logs
- Recolectar información vital sobre el sistema cuando ocurren problemas.

Percona asegura que esta suite de herramientas ha sido fuertemente testeada y su efectividad está comprobada por la propia experiencia de Percona al diseñarla en conjunto con su aplicación en varias aplicaciones web de gran porte.

MySQL Report [XVI] genera una vista amigable de datos importantes del estado de MySQL. MySQL Report transforma valores de la utilidad SHOW STATUS en un reporte sencillo de interpretar con el objetivo de prever una descripción detallada del estado de MySQL. Es la mejor alternativa a interpretar SHOW STATUS manualmente.

mysqlsla [XVII] Esta herramienta analiza y procesa un conjunto de archivos de logs de MySQL, especialmente útil para obtener información del archivo de registro de consultas lentas (slow_queries) Ejemplos:

- Slow log: mysqlsla -lt slow slow.log
- General log: mysqlsla -lt general general.log
- Binary log: mysqlbinlog bin.log — mysqlsla -lt binary -

2.4.2. Herramientas para PostgreSQL

Debido a su popularidad PostgreSQL cuenta también con un buen número de herramientas de apoyo al administrador de sistema y DBA.

2.4.2.1. Herramientas de Administración

PostgreSQL cuenta con herramientas de monitoreo libres particularmente debido a su buena integración con los sistemas operativos de la familia Unix. Se verán algunas de ellas.

pgAdmin III [XVIII] permite administrar con facilidad la mayor parte de las tareas administrativas rutinarias. Está disponible en forma libre bajo los términos de la Licencia

[XV] http://www.percona.com/software/percona-toolkit
[XVI] http://hackmysql.com/mysqlreport
[XVII] http://hackmysql.com/mysqlsla
[XVIII] http://www.pgadmin.org/

PostgreSQL. Se enumeran las principales tareas que pueden realizarse desde la interfaz provista por pgAdmin III

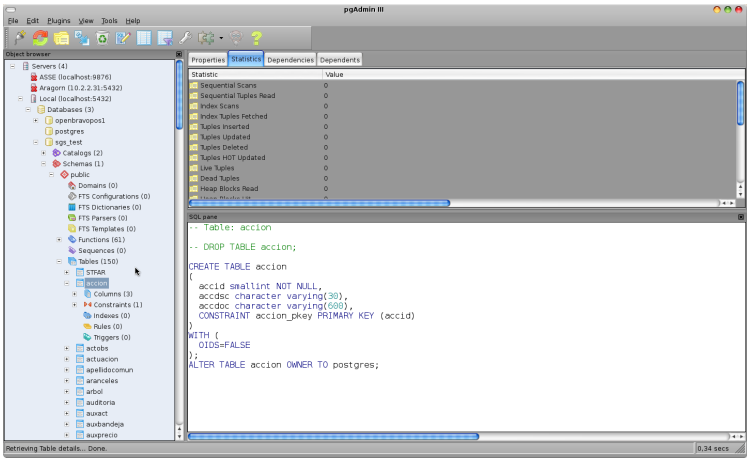

Figura 2.11: Captura de pantalla: Ventana principal

1. **"Main window"** muestra la estructura de la base de datos, y todos los detalles de los objetos en ella. Ver fig. 2.11
2. **"Add server"** para agregar una conexión a una nueva base de datos.
3. **"Change password"** para cambiar la contraseña del usuario que está actualmente conectado con la base de datos seleccionada.
4. **"Server control"** permite controlar el estado de los servidores de base de datos, iniciar y detener sus actividades. Ver fig. 2.12

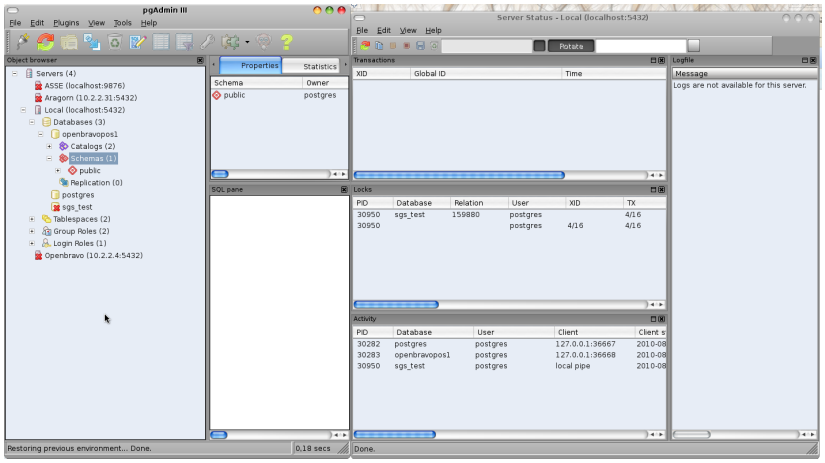

Figura 2.12: Captura de pantalla: Control del servidor, conexiones de usuarios, transacciones, bloqueos y regristros.

5. **"Debugger"** permite la depuración de pl/pgsql y edb-spl y funciones o *stored procedures*

6. **"Export tool"** permite la exportación de datos de la herramienta de consulta.
7. **"Edit grid"** permite la visualización y edición de datos en una tabla o vista seleccionada.
8. La herramienta **"Maintenance"** ejecuta la tarea de reconstruir las estadísticas sobre las bases de datos y tablas, limpia datos sin utilizar y reorganiza índices.
9. La herramienta **"Backup"** llama a la herramienta de volcado (*dump*) de PostgreSQL para crear archivos de copia de seguridad de sus datos.
10. La herramienta **"Restore"** llama a la herramienta de restauración de PostgreSQL para restaurar los datos de los archivos de copia de seguridad.
11. **"Grant Wizard"** permite la asignación de privilegios de grupo o usuario en muchos objetos a la vez.
12. La herramienta **"Report tool"** genera informes sencillos a partir de datos y objetos de las bases de datos.
13. La ventana **"Server status"** muestra los usuarios conectados en el momento, los bloqueos, las transacciones preparadas y el registro del servidor de base de datos seleccionado.

PostgreSQL Maestro [XIX] es otra herramienta de administración comercial para PostgreSQL. Algunas de las características ofrecidas son:

- Soporte para todas las versiones de PostgreSQL a partir de la 7.3
- Administración sencilla de objetos de la base de datos
- Herramienta de diseño de base de datos
- Debugger de PL/pgSQL
- Interfaz amigable de acceso a las características de seguridad de PostreSQL
- Editor de SQL
- Constructor de consultas visual con soporte para subconsultas y sentencias UNION
- Conexión a servidores PostgreSQL remotos vía SSH o túneles HTTP
- Importación/Exportación de datos desde/hacia los formatos más populares
- Poderoso editor de BLOB

La herramienta provee también un conjunto de herramientas para editar y ejecutar *scripts* de SQL, construir diagramas visuales para datos numéricos y componer cubos OLAP.

[XIX] http://www.sqlmaestro.com/products/postgresql/maestro

2.4.2.2. Herramientas de Monitoreo

PG_TOP[XX] es un software de consola para al estilo del popular comando "top" the Unix. De manera similar a "top", pg_top permite monitorear los procesos de PostrgreSQL. Algunas de las vistas disponibles:

- sentencia SQL en ejecución
- plan de ejecución de la sentencia SQL en ejecución
- bloqueos
- estadísticas de tablas
- estadísticas de índices.

PGFouine[XXI] es un analizador de los logs de PostgreSQL. Provee una clara visión de la actividad de la base de datos y puede generar reportes con consultas más frecuentes y de baja performance. Está escrito en PHP, tiene licenciamiento libre y está diseñado para analizar archivos de logs de gran tamaño con poco consumo de la memoria de la computadora.

Postgres Enterprise Manager[XXII] ofrecida por EnterpriseDB, esta herramienta ofrece posibilidades de monitoreo más amplias que las herramientas libres.

Algunas de las métricas que se ofrecen son:

- Rendimiento
  - Consultas y transacciones lentas
  - Transacciones detenidas
  - Tiempo desde el último Vacuum / Analyze
- Mediciones
  - Commits / Rollbacks / por minuto
  - Bloques leídos por minuto
  - Archivos WAL
  - Filas insertadas, actualizadas y borradas por minuto
- Utilización
  - Conexiones y locks
  - Edad y tamaño de la base de datos
  - Roles de Login/Group/Super
  - Cantidad de bases y esquemas

[XX] http://ptop.projects.postgresql.org
[XXI] http://pgfoundry.org/projects/pgfouine
[XXII] http://www.enterprisedb.com/products-services-training/products/postgres-enterprise-manager

- Cantidad de tablas, vistas y funciones
- Cantidad de secuencias
- Cantidad de transacciones preparadas
- Cantidad de tuplas devueltas, insertadas, actualizadas y borradas
- Escaneos secuenciales y de índices por minuto

2.4.3. Herramientas "Multi Motor"

A continuación se presentarán herramientas de administración genéricas que pueden ser conectadas a diferentes sistemas de acuerdo a la disponibilidad de conectores.

SQuirreLSQL [XXIII] es un cliente gráfico escrito en Java que permite visualizar la estructura de cualquier base de datos compatible con JDBC, visualizar las tablas y ejecutar consultas SQL entre otras cosas.
Utiliza JDBC para interactuar con los diversos tipos de bases de datos. Dado que la misma fue desarrollada en Java, esta aplicación debe correr en cualquier plataforma que tenga una JVM. Esta aplicación es distribuida bajo licencia GNU.

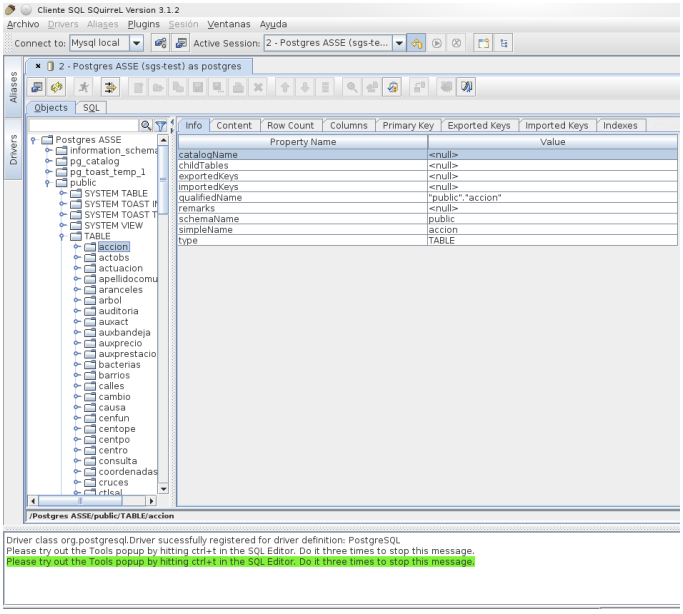

Figura 2.13: Captura de pantalla: Visualización de propiedades de tabla.

La figura 2.13, muestra la forma en que SQuirrel despliega la información referente a una tabla. Las funcionalidades de SQuirreL pueden extenderse mediante el uso de *plugins*. SQuirreL fue liberado originalmente bajo la licencia GPL. Desde la versión 1.1beta2 ha sido liberado bajo la licencia LGPL[34].

RazorSQL[XXIV] es una herramienta de administración de base de datos y editor de SQL para Windows, Mac OS X, Linux, y Solaris. Ha sido testeada con más de 30 bases

[XXIII] http://www.squirrelsql.org
[XXIV] http://www.razorsql.com

de datos utilizando JDBC u ODBC. Cuenta por supuesto con soporte para MySQL y PostgreSQL.

Características:

- navegador de la base de datos
- editor de SQL
- herramientas visuales para efectuar las operaciones habituales sobre tablas, índices, vistas y secuencias
- herramientas visuales para administrar *stored procedures*, funciones y *triggers*
- editor de programación incorporado
- importación/exportación de datos en varios formatos
- herramienta de comparación de tablas
- herramienta de búsqueda en base de datos
- interfaz de línea de comandos

RazorSQL provee de herramientas visuales y características avanzadas para permitir al usuario navegar, administrar y programar para la base de datos en forma sencilla.

2.4.4. Herramientas de análisis y monitoreo del Sistema Operativo

El funcionamiento del sistema manejador de base de datos está estrechamente relacionado con el desempeño del sistema operativo. El DBA debe estar interesado en las diferentes herramientas que permiten monitorear el estado de los procesos, principalmente el consumo de memoria, paginado y uso de memoria virtual.
En muchos casos los servidores de bases de datos sufren de agotamiento de recursos de hardware tales como sobrecarga de la CPU, *memory swapping* demasiado frecuente y lentitud de procesamiento de I/O. Es improbable que exista una única herramienta que pueda analizar el estado general del sistema, el desafío se encuentra en entender cada una de las herramientas disponibles para extraer la información adecuada de cada una de ellas. A continuación de mencionarán algunas de estas utilidades

filefrag [xxv] permite obtener información sobre la fragmentación de archivos. En archivos de gran tamaño, el sistema de archivo -incluso del tipo ext2, ext3 y ext4-sufren de fragmentación afectando el rendimiento del sistema. El resultado de ejecutar filefrag sobre un archivo es la cantidad de "extents", reflejando de esta manera su estado. Un 1 indicará que el archivo no está fragmentado, mientras que un número alto (por ejemplo 60), mostrará un posible factor de degradación de rendimiento. Ver figura 2.14

[xxv] http://linux.die.net/man/8/filefrag

```
root\# filefrag /var/lib/mysql/mysql/db.frm  -v
Filesystem type is: ef53
File size of /var/lib/mysql/mysql/db.frm is 9582 (3 blocks, blocksize 4096)
 ext logical physical expected length flags
   0       0  7740957               3 eof
/var/lib/mysql/mysql/db.frm: 1 extent found
```

Figura 2.14: Ejecución del comando filefrag con la opción "verbatim"

vmstat [XXVI] es herramienta provista por los sistemas operativos basados en UNIX, permite monitorizar el estado del hardware donde se encuentra corriendo la base de datos, como lo es disco, memoria, procesos y demás partes. Ver figura 2.15

```
virtualstudents:~ # vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- -----cpu------
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 605140 314424 1503180    0    0     0     2    1    2  4  1 95  0  0
```

Figura 2.15: Ejecución del comando vmstat

iostat [XXVII] esta herramienta nos permite observar el rendimiento obtenido por los dispositivos de almacenamiento como los discos físicos. Ver figura 2.16

```
virtualstudents:~ # iostat
Linux 2.6.31.14-0.4-desktop (virtualstudents)  05/31/12  _x86_64_ (2 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          0.10    3.87    1.04    0.08    0.00   94.91

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda               0.24         0.13         2.71     856594   17217720
sdb               0.97         1.17         4.57    7432341   29039999
```

Figura 2.16: Ejecución del comando iostat

2.5. Resumen

A la hora de optar por un RDBMS FOSS para utilizarlo como *backend* de una aplicación, existen una serie de elementos a tener en cuenta para decidir.

Algunos factores son propios del motor, como los métodos elegidos para el almacenamiento de los datos, la tolerancia a fallas, las posibilidades de respaldo y las funcionalidades ofrecidas.

[XXVI] http://linuxcommand.org/man_pages/vmstat8.html
[XXVII] http://linuxcommand.org/man_pages/iostat1.html

Otros de estos factores, son externos. Deberán considerarse aquí la disponibilidad de herramientas y las posibilidades que estas ofrecen tanto para monitorear el estado del servidor como para corregir eventuales fallas.

Como se ha visto a lo largo del capítulo, las herramientas presentadas ofrecen un número de funcionalidades para la administración. De ellas, el DBA deberá seleccionar aquella que se ajuste más a sus necesidades. Así, si se trata de administrar un RDBMS MySQL, puede hacerse uso de MySQL Workbench, pero si se desean administrar diversos motores con una única herramienta, posiblemente SquirrelSQL sea una mejor elección. De igual manera, la opción por una herramienta comercial queda sujeta a la necesidad de alguna de las funcionalidades extras que ellas ofrecen.

Merece también una nota especial el soporte ofrecido por el/los proveedor(es). Una diferencia fundamental entre los motores comerciales y los motores FOSS es que el soporte está dado por una comunidad de desarrolladores y usuarios y no por una empresa comercial. Este aspecto ofrece la ventaja de poder acceder - ante un problema dado - a información de primera mano de aquellos usuarios que han enfrentado problemas similares o de contactar al desarrollador que programó el módulo problemático .

A los efectos de este proyecto, el motor de partida es Oracle que cumple los requerimientos del SGS. En este capítulo, se ha mostrado que tanto MySQL como PostgreSQL también satisfacen los requerimientos mencionados en la sección 2.2. MonetDB, fue preseleccionado debido a que el paradigma de almacenamiento columnar promete una mejora en el rendimiento pero debió ser descartado en forma prematura debido a la imposibilidad de integración con Genexus.

Capítulo 3

Ajustes de Configuración y Pruebas de Procedimientos Relevantes

3.1. Introducción

En este capítulo se detalla la investigación y prueba acerca de los siguientes procedimientos relevantes para ASSE:

1. Ajustes de configuración
2. Respaldos en caliente
3. Particionamiento
4. Concurrencia
5. Optimización de consultas

3.2. Ajustes de configuración

Una vez instalado el servidor con una configuración básica, podría ser necesario realizar ajustes para mejorar el rendimiento o facilitar los procedimientos de mantenimiento. No se trata de comparar posibilidades de optimización entre MySQL y PostgreSQL, sino de comprobar si es posible en ambos, a grandes rasgos, realizar ajustes del sistema si fuera requerido.

A continuación de detallan las posibilidades de configuración de cada uno de los RDBMS considerados.

3.2.1. MySQL

MySQL provee un conjunto de opciones y variables que permite manejar los buffer y cache según las necesidades. El archivo de configuración que contiene estas variables para el ambiente se encuentra usualmente en "/etc/mysql/my.cfg". Para diferentes distribuciones puede variar la ubicación pero el nombre del archivo sigue siendo el mismo.
Los ajustes a realizar, requieren un monitoreo del estado del servidor en todo momento. Para ello se cuenta con comandos específicos que permiten obtener la información. El

comando “mysqladmin” con el atributo “variables” o desde una sesión en MySQL con la sentencia “SHOW VARIABLES” permiten visualizar los valores de las variables con el cual el servicio se inició.

A continuación se detallarán algunas de las variables más relevantes a la hora de realizar una optimización. [15]

La variable **table_cache**, permite definir la cantidad máxima de tablas que se contendrán en cache y que se están accediendo. Cada vez que se accede a una tabla se la intentará ubicar en memoria, y en caso de haber alcanzado el cupo máximo se quitará una tabla que ya no este siendo accedida.

La cantidad de tablas abiertas en un mismo instante puede ser mayor a la cantidad de tablas definido en el esquema esto se debe a que MySQL utiliza múltiples subprocesos para atender las consultas realizadas. De esta manera los accesos son mas rápidos que cuando son accedidos desde el disco.

Esta variable se estableció en un valor de 512, no es un valor definitivo debido a que se debe ajustar de acuerdo a un ambiente de producción observado los datos correspondientes a la cantidad de tablas abiertas en el momento de mayor consumo de recursos por parte de la aplicación.

La variable **key_buffer_size** repercute en el espacio de memoria asignado a los *buffers* de los índices. Cuanto más grande sea este valor, más cantidad de índices podrán alojarse en memoria, por lo cual los accesos y búsquedas en los índices serán más rápidos. El valor establecido para esta variable denota la cota máxima a ser utilizada por el sistema, la cual no es reservada desde el inicio sino que es asignada a medida que sea requerida. Una buena práctica a los efectos de asignar el tamaño de memoria es realizar una comparación entre los valores de estado “key-read-requests” y “key reads”. El primero indica la cantidad de búsquedas en el cache, mientras que el segundo la cantidad de búsquedas en el disco físico, una buena relación es de 1 a 100.

La variable **query_cache_size** determina la cantidad de memoria asignada para la cache de consultas, básicamente MySQL almacena en cache cada consulta y luego si existe una consulta exactamente igual, devuelve el valor almacenado sin necesidad de realizar nuevamente todo el proceso. Si se modifica, borra o ingresa algún dato de una tabla, automáticamente se borran todas las consultas en cache que acceden a esa tabla. Esta variable se estableció en un valor de 100MB debido a que existen muchas consultas que son repetitivas y existen tablas que no tienen muchas cambios como es el ejemplo de la tabla de funcionarios.

La variable **log-slow-queries** permite configurar un registro de las consultas que demoran cierta cantidad de tiempo. Mediante la opción **long_query_time** se puede especificar la cantidad de tiempo mínimo que deben demorar las consultas para ser registradas. Esto brinda una visión de cuáles son las consultas que generan demora ya sea por problemas de índices o por falta de optimización.

Hasta el momento se mostró la configuración de variables genéricas de MySQL, pero debido a que en la migración del SGS utilizará el motor InnoDB, se deben realizar

configuraciones especificas para este motor.

La variable **innodb_buffer_pool_size** define el tamaño del buffer de memoria que InnoDB emplea para el almacenamiento intermedio de los datos e índices de sus tablas. Un incremento en el valor asignado a esta variable, permite bajar drásticamente el tiempo de operaciones de entrada/salida, pero se debe tener cuidado debido a que un exceso de la memoria asignada a este buffer puede causar que el sistema operativo comience a paginar. Se entiende que una buena medida es el 70 % o 80 % de la memoria para un servidor exclusivo y dedicado a MySQL. En el entorno utilizado se le asignó a esta variable 1 GB de memoria.

A partir de la declaración de la variable **innodb_datahome_dir** se define el directorio donde se alojarán los data_files pertenecientes a las tablas de InnoDB.

La variable **innodb_lock_wait_timeout** permite definir el tiempo que debe esperar una transacción por un bloqueo "*DeadLock*" antes que el DBMS la cancele para ello InnoDB automáticamente detecta los bloqueos.

La opción **innodb_file_per_table** provoca que InnoDB cree cada nueva tabla utilizando su propio fichero .ibd para almacenar datos e índices, en lugar de colocarlo en el espacio de tablas compartidas.

3.2.2. PostgreSQL

La opción **max_connections** = $< num >$ - establece el número máximo de clientes que pueden conectarse a la base de datos en un momento dado.

shared_buffers = $< num >$ - Determina la cantidad de memoria que Postgres utiliza para el caché de datos. Se estima como valor adecuado, alrededor de un 25 % de la memoria RAM disponible en el sistema. En términos generales, niveles de RAM inferiores a 1GB exigen que el valor del parámetro sea inferior al 25 %. Ante una mayor cantidad de memoria RAM, incrementar el porcentaje de memoria asignada a Postgres no parece tener mayor incidencia.

effective_cache_size = $< num >$ - Este valor será utilizado por el optimizador de PostgreSQL para determinar si la ejecución de una consulta cabe en memoria RAM o no. El optimizador, se valdrá de este valor para determinar la ventaja en el uso de los índices. Cuanto mayor sea el valor de este parámetro, más aumentará la probabilidad de la utilización de un índice. Se estima como una configuración conservadora, establecerlo en un 50 % de la memoria total del sistema y alrededor de un 75 % como una configuración más agresiva.

work_mem = $< num >$ - Esta opción se utiliza para controlar la cantidad de memoria en uso en las operaciones de ordenación (*sorting*). Este parámetro no opera sobre el sistema, sino sobre cada operación de orden, por lo tanto una consulta compleja que requiera ordenamiento sobre 5 tablas, requerirá una RAM equivalente a 5 veces el valor establecido por work_mem.

autovacuum, max_fsm_pages = $< num >$ - Estas opciones ayudan a controlar espacio marcado como libre. Cuando se elimina de la tabla un registro, éste es marcado como "libre" y no se elimina del disco inmediatamente. Este espacio puede ser reutilizado para nuevas inserciones en la tabla. Si la configuración tiene una alta tasa de operaciones

de "*DELETE*" y de "*INSERT*", puede ser necesario aumentar este valor para evitar el crecimiento del espacio de la tabla en disco.

fsync = < *boolean* > - Esta opción determina si todas las páginas sincronizan los archivos WAL de realizar el "*commit*" de la transacción. Si no está habilitada la opción "fsync" existe la posibilidad de corrupción de datos irrecuperable.

random_page_cost = < *num* > - sugiere al optimizador el costo de la búsqueda secuencial de una página. Un valor más alto hace que sea más probable una búsqueda secuencial y, por lo tanto, será más útil con discos rápidos[33].

3.3. Respaldos físicos

Los respaldos físicos forman parte clave de los requerimientos del SGS. El volumen de datos manejados hace que se necesite contar con la posibilidad de realizar respaldos en tiempo razonable y que los mismos puedan hacerse con la base de datos *online*. Se presentan a continuación las herramientas evaluadas para los DBMS considerados.

3.3.1. Respaldo en MySQL: XtraBackup

Percona XtraBackup[23] es un conjunto de herramientas Open Source desarrollada por la empresa Percona para realizar respaldos de la base de datos MySQL. XtraBackup opera sobre InnoDB y MyIsam, en esta sección se mencionará únicamente InnoDB:

Las principales características son:

- Respaldos en caliente sin realizar bloqueos
- Respaldos completos o parciales
- Respaldos en paralelo
- Respaldos incrementales
- Compresión de respaldos

Se puede decir que Percona XtraBackup es una composición de dos herramientas: *xtrabackup* y *innobackupex*. La primera de ella desarrollada en lenguaje C, permite exclusivamente realizar respaldos de motores basados en InnoDB, mientras que la segunda es un *script* Perl [I].

XtraBackup es la alternativa libre a *innobackup*[II] distribuida por Oracle.

Existen diversas formas de instalar los ejecutables de la herramienta de acuerdo al tipo de sistema operativo que se utilice, por lo que en este caso, dado que la distribución utilizada es OPEN SUSE, se optó por utilizar el instalador RPM. La instalación es sencilla y no tiene mayores complicaciones.

Una vez instalado los ejecutables, se debe configurar el archivo "my.cnf" de la base de datos MySQL agregando una nueva sección "[xtrabackup]" en la cual debe especificarse mediante la variable "target_dir" el directorio donde se guardarán los respaldos. Este

[I] http://www.percona.com/doc/percona-xtrabackup/innobackupex/innobackupex_script.html?id=percona-xtrabackup:innobackupex:start

[II] InnoDB Hot Backup es una herramienta comercial, más detalles en http://www.innodb.com/wp/products/hot-backup/

valor de configuración no es un requerimiento excluyente debido a que se puede especificar el destino del respaldo desde la línea de comando.

El principal problema de realizar un respaldo en caliente (*Hot Backup*) es que mientras se está haciendo una copia de la base de datos, otro cliente del DMBS puede introducir modificaciones, normalmente ingresos, modificaciones o eliminaciones de registros. Este hecho puede generar una inconsistencia a la hora de restaurar los datos.

Respaldo

XtraBackup hace uso de los "redo log" [7] de InnoDB encargados de mantener el registro de cambios realizados al motor de almacenamiento. Básicamente se podría explicar el funcionamiento de la siguiente manera, cuando se inicia el proceso de respaldo, XtraBackup guarda el valor LSN (Numero de Secuencia del Log) y realiza el copiado de los archivos que contiene los datos. El siguiente paso es reacondicionar los datos, para lo cual se utiliza el valor LSN y los archivos "redo log", aplicando ajustes de acuerdo a los cambios registrados después del valor LSN.

La realización de un respaldo se ejecuta en forma muy sencilla mediante la ejecución desde una consola del comando "xtrabackup" con el parámetro "–backup". En el escenario de evaluación, es necesario tomar los tiempos de ejecución del respaldo, utilizando el comando "time", ejemplificado en la figura 3.1

```
virtualstudents:/mysql# time xtrabackup --backup --datadir=/varios/02/restore/mysql/
                        --target-dir=/varios/02/backup/mysql/
xtrabackup  Ver 1.2 Rev 132 for 5.1.45 pc-linux-gnu (i686)
xtrabackup: uses posix_fadvise().
xtrabackup: cd to /varios/02/restore/mysql/
xtrabackup: Target instance is assumed as followings.
xtrabackup:   innodb_data_home_dir = ./
xtrabackup:   innodb_data_file_path = ibdata1:10M:autoextend
xtrabackup:   innodb_log_group_home_dir = ./
xtrabackup:   innodb_log_files_in_group = 2
xtrabackup:   innodb_log_file_size = 5242880
>> log scanned up to (479233325124)
Copying ./ibdata1
     to /varios/02/backup/mysql//ibdata1
>> log scanned up to (479233325124)
>> log scanned up to (479233325124)
.................................
.................................
>> log scanned up to (479233325124)
        ...done
xtrabackup: The latest check point (for incremental): '479233325124'
>> log scanned up to (479233325124)
xtrabackup: Stopping log copying thread.................
xtrabackup: Transaction log of lsn (479233325124) to (479233325124) was copied.

real 31m48.617s
user 11m14.601s
sys 5m24.839s
```

Figura 3.1: Ejecución del comando "xtrabackup" con parámetro "backup"

En la figura 3.1 se puede observar que la realización del respaldo completo de la base de datos insumió un tiempo de aproximadamente 32 minutos. Es importante resaltar que dicho respaldo se realizó sin compresión.
Una vez realizado el respaldo y a los efectos de restaurar la base de datos, es necesario

realizar un tratamiento a los archivos de respaldo, por lo que se debe ejecutar el comando "xtrabackup" nuevamente pero esta vez utilizando el parámetro "–prepare".

```
virtualstudents:/varios/02/backup # time xtrabackup --prepare --target-dir=/varios/02/backup/mysql/
xtrabackup  Ver 1.2 Rev 132 for 5.1.45 pc-linux-gnu (i686)
xtrabackup: cd to /varios/02/backup/mysql/
xtrabackup: This target seems to be not prepared yet.
xtrabackup: xtrabackup_logfile detected: size=2097152, start_lsn=(479233325124)
xtrabackup: Temporary instance for recovery is set as followings.
xtrabackup:   innodb_data_home_dir = ./
xtrabackup:   innodb_data_file_path = ibdata1:10M:autoextend
xtrabackup:   innodb_log_group_home_dir = ./
xtrabackup:   innodb_log_files_in_group = 1
xtrabackup:   innodb_log_file_size = 2097152
xtrabackup: Starting InnoDB instance for recovery.
xtrabackup: Using 104857600 bytes for buffer pool (set by --use-memory parameter)
InnoDB: The InnoDB memory heap is disabled
InnoDB: Mutexes and rw_locks use InnoDB's own implementation
InnoDB: Warning: innodb_file_io_threads is deprecated. Please use innodb_read_io_threads
        and innodb_write_io_threads instead
110802 22:17:35  InnoDB: highest supported file format is Barracuda.
InnoDB: The log sequence number in ibdata files does not match
InnoDB: the log sequence number in the ib_logfiles!
110802 22:17:35  InnoDB: Database was not shut down normally!
InnoDB: Starting crash recovery.
InnoDB: Reading tablespace information from the .ibd files...
InnoDB: Last MySQL binlog file position 0 502505764, file name ./mysql-bin.000119
110802 22:17:36 InnoDB Plugin 1.0.6-unknown started; log sequence number 479233325124

[notice (again)]
  If you use binary log and don't use any hack of group commit,
  the binary log position seems to be:
InnoDB: Last MySQL binlog file position 0 502505764, file name ./mysql-bin.000119

xtrabackup: starting shutdown with innodb_fast_shutdown = 1
110802 22:17:36  InnoDB: Starting shutdown...
110802 22:17:37  InnoDB: Shutdown completed; log sequence number 479233325134

real    0m2.017s
user    0m0.112s
sys     0m0.060s
```

Figura 3.2: Ejecución del comando "xtrabackup" con parámetro "prepare"

Restauración

Una vez que el respaldo es tratado (figura 3.2), queda listo para ser restaurado. Para ello, alcanza con copiar los archivos generados al directorio que contienen los archivos de la base de datos definido en la configuración de MySQL y luego iniciar el servicio, de esta manera queda completado el proceso de restauración de un respaldo.

3.3.2. Respaldo en PostgreSQL: pg_hotbackup

Pg_hotbackup es un *script* escrito en Perl cuya función es la automatización del proceso de respaldo "en caliente" [30]. En resumen opera realizando tres pasos fundamentales:

- Marca el inicio de la copia
- Copia los archivos de la base de datos
- Cierra el respaldo copiando los archivos de log binario

Además de esto, permite diferentes niveles de compresión, utilizando gzip como compresor. Con los parámetros adecuados permite cambiar el destino de la copia, etiquetarla, ajustar el nivel de compresión, etc. Para la realización del respaldo físico, se explotan los archivos de bitácora (WAL) que PostgreSQL mantiene en el directorio pg_xlog. En estos archivos, se registra cada cambio realizado a los archivos de datos de la base con el objetivo de regresar a una situación estable en caso de caída de la base. La estrategia a utilizar combina una copia física de los archivos de datos con una copia de seguridad de los archivos WAL.

Con esto, no es necesario tener una copia consistente de los datos ya que en el proceso de restauración cualquier inconsistencia será corregida por los archivos de bitácora. Las utilidades corrientes de archivado como "tar" o "cpio" serán suficientes para realizar el respaldo. El respaldo físico solo soporta la restauración de un *cluster* completo y no de un subconjunto.

Configuración del archivado WAL

En sentido abstracto, PostgreSQL produce una secuencia indefinida de registros de WAL que se almacenan en archivos con nombres numéricos que reflejan el orden de secuencia. Para garantizar el éxito del respaldo, alcanza con que existan registros de bitácora previamente a comenzar el respaldo físico. Para habilitar el archivado WAL, se deben configurar dos parámetros en el archivo postgres.conf.
Estos son:

- **archive_mode** que debe tener el valor "on"
- **archive_command** que debe tener el comando que se utilizará para copiar los archivos WAL.

Un posible valor para archive_command es :

```
cp -i %p /<directorio_destino>/%f </dev/null'
```

Dado que %p será sustituido por el camino completo a cada uno de los archivos WAL y %f por su nombre, el comando anterior creará una copia de cada archivo WAL en el directorio de destino.

Este comando de archivo se ejecutará bajo la propiedad del usuario de la base de datos, por lo que deberá tener los permisos necesarios en el directorio de destino.Para evitar la sobreescritura indeseada de respaldos anteriores, el comando anterior puede ser mejorado con algo así como:

```
archive_command = 'test ! -f /<dir_destino>/%f && cp %p /<dir_destino>/%f'
```

Resumen del procedimiento de respaldo utilizando el script pg_hotbackup

Una vez realizada la configuración necesaria para el funcionamiento de los archivos WAL, el procedimiento para hacer una copia de seguridad de la base de datos se reduce a la ejecución del comando "pg_hotbackup" que admite las siguientes opciones:

```
-l <s>, --label=<s>       Use label <s> for backup file
-b <s>, --backupdir=<s>   Write backup file to directory <s>
-D <s>, --datadir=<s>     Use postgresql data directory <s>
-Z <i>, --compress=<i>    Compress files with compression level <i>
-U <s>, --username=<s>    Connect as user <s>
```

```
-W, --password           Prompt for password
-p <i>, --port=<i>       Use port <i>
-c <s>, --command=<s>    Use command <s> to perform backup
```

Con esto pg_hotbackup creará un archivo "label.tar.gz" en el directorio "backupdir" y copiará los archivos WAL según el comando especificado en la configuración de archive_command.

Restauración

Para la restauración de la base, el primer paso es detener el servidor, si es que se está ejecutando. Luego, se descomprime el archivo de respaldo sustituyendo al *cluster* de datos original y finalmente se copian los archivos WAL dentro del directorio pg_xlog dentro del *cluster* de datos. Si este directorio existe, debe borrarse antes de copiar los archivos WAL. En caso de no existir, debe crearse para realizar la copia de los archivos. Finalmente, puede reiniciarse el servidor de la base de datos.

Notas en la documentación y pruebas realizadas

La primera prueba de respaldo físico se realizó enviando el archivo de respaldo a un segundo disco a los efectos de no generar problemas de desempeño. Utilizando un nivel 5 de compresión con el algoritmo gzip, el archivo resultante obtenido fue de 6.9GB, para un tamaño original de 35GB y el respaldo tardó 37 minutos en completarse.

En la documentación se advierte que las operaciones sobre índices *hash* no se registra en los archivos WAL. Se desaconseja especialmente la ejecución de comandos CREATE DATABASE y CREATE TABLESPACE durante la realización de los respaldos.

Como primera prueba, se ensayó la inserción de datos y la modificación de estructuras de tablas durante el respaldo. Todos estos cambios se realizaron antes de la copia de los archivos WAL y, por tanto, se vieron reflejados al levantar el respaldo[3].

La segunda prueba realizada, consistió en la realización de un respaldo mientras dos tablas diferentes estaban siendo bloqueadas y ambas mantenían transacciones abiertas. El respaldo finalizó sin reportar problemas y al levantar los datos respaldados no se registró ninguna señal de las transacciones que no habían sido confirmadas.

3.4. Particionamiento

El particionamiento es en gran medida una técnica de optimización que pretende mejorar los tiempos de respuesta de las consultas, así como el mantenimiento de las tablas, y que puede ser especialmente útil en tablas muy grandes. Teóricamente, el particionado puede realizarse de manera horizontal, alojando distintas filas de una misma tabla en distintas particiones físicas, o de manera vertical, alojando distintas columnas en distintas particiones. En esta sección se hará referencia exclusivamente al particionamiento horizontal.

Básicamente, el particionamiento se realiza utilizando una función de particionado, que determina en qué partición de las existentes en la tabla van a residir los datos que se insertan. Cada partición además puede tener sus propias propiedades de almacenamiento. Las tablas particionadas pueden ser mostradas por el DBMS como una única tabla,

realizando el sistema la gestión automática de lectura y escritura en cada una de las particiones.

3.4.1. Tipos de Particionamiento

Existen diversas funciones de particionado, dependiendo de cada DBMS cuales implementa y de que manera. A continuación se listan las principales funciones de particionamiento:

- **Por Rango**, se asigna a cada partición un rango de valores, los rangos no deben solaparse ya que sino se generaría una inconsistencia en el almacenamiento de los datos.

- **Por Hash**, se utiliza para conseguir una distribución homogénea de los datos en las particiones. Debemos asignar una función de *hash* interna la cual definirá en que partición se inserta, consulta o actualiza un dato.

- **Por Lista**, se define de forma explícita a través de una lista los valores que la clave de particionado debe tomar para asociarlo a la tabla particionada

3.4.2. Ventajas del particionado de tablas

Las principales ventajas del particionado de tablas radican en la mejora del rendimiento en varios aspectos, a saber:

1. **Uso del sistema de archivos**: una tabla particionada puede alojar más datos que los disponibles en un único sistema de archivos. Esto, permite entre otras cosas una mayor flexibilidad a la hora de escalar la base de datos,

2. **Inserción y borrado de datos**: borrar datos comunes a una partición, evita la necesidad de recorrer una tabla buscando la información a remover. En lugar de esto, puede eliminarse la partición correspondiente al juego de datos. Análogamente, la inserción de datos puede verse facilitada añadiendo nuevas particiones para el almacenamiento de datos específicos,

3. **Optimización de consultas**: una consulta será más eficiente si, en lugar de tener que recorrer una gran tabla buscando el juego de datos que satisfacen una condición dada, puede devolver una partición entera realizada con dicha condición o buscando en una partición en lugar de en la tabla entera. Como ejemplo, si se particiona una tabla de ventas en la que se almacenan las transacciones de distintas tiendas, particionando por el identificador de la tienda, puede accederse a las ventas de una tienda particular consultado solamente la partición correspondiente a dicha tienda.

Se muestra a continuación la técnica de particionado en cada uno de los DBMS evaluados.

3.4.3. Particionado en MySQL

En MySQL[2], el particionado de tablas permite distribuir porciones de datos de las tablas en un sistema de archivos de acuerdo a reglas definidas por el usuario. Estas reglas, llamadas funciones de particionado, permiten separar una tabla lógica en varios archivos físicos dividiendo según clave, rangos, listas de valores o funciones de *hash* entre otras. La división por clave es similar al particionado por *Hash*, con la diferencia que el usuario brinda la función de *hash* utilizada.

El objetivo principal del particionado es reducir la cantidad de accesos físicos al medio de almacenamiento cuando se ejecuta una consulta.

3.4.3.1. Qué se necesita para utilizar el particionado de tablas con MySQL

El único requisito necesario para utilizar el particionado es que los ejecutables de MySQL estén compilados con soporte para particionado. Los ejecutables de la versión Community de MySQL que distribuye Sun Microsystems incluyen el soporte para particionado. Para comprobarlo, se utiliza el comando de MySQL SHOW VARIABLES:

```
mysql> SHOW VARIABLES LIKE '%partition%';
+-------------------+-------+
| Variable_name     | Value |
+-------------------+-------+
| have_partitioning | YES   |
+-------------------+-------+
1 row in set (0.01 sec)
```

En versiones anteriores a MySQL 5.1.6, la variable have_partitioning recibía el nombre have_partition_engine. También puede utilizarse el comando "SHOW PLUGINS" (debe observarse la línea "partition"):

```
mysql> SHOW PLUGINS;
+------------+----------+----------------+-----------------+---------+
| Name       | Status   | Type           | Library         | License |
+------------+----------+----------------+-----------------+---------+
| binlog     | ACTIVE   | STORAGE ENGINE | NULL            | GPL     |
| partition  | ACTIVE   | STORAGE ENGINE | NULL            | GPL     |
| CSV        | ACTIVE   | STORAGE ENGINE | NULL            | GPL     |
| MEMORY     | ACTIVE   | STORAGE ENGINE | NULL            | GPL     |
| InnoDB     | ACTIVE   | STORAGE ENGINE | NULL            | GPL     |
| MyISAM     | ACTIVE   | STORAGE ENGINE | NULL            | GPL     |
| MRG_MYISAM | ACTIVE   | STORAGE ENGINE | NULL            | GPL     |
| ndbcluster | DISABLED | STORAGE ENGINE | NULL            | GPL     |
| ARCHIVE    | ACTIVE   | STORAGE ENGINE | ha_archive.so   | GPL     |
| FEDERATED  | DISABLED | STORAGE ENGINE | ha_federated.so | GPL     |
| BLACKHOLE  | ACTIVE   | STORAGE ENGINE | ha_blackhole.so | GPL     |
| EXAMPLE    | ACTIVE   | STORAGE ENGINE | ha_example.so   | GPL     |
+------------+----------+----------------+-----------------+---------+
12 rows in set (0.00 sec)
```

3.4.3.2. Creación de particiones

Para la creación de particiones, se utiliza el modificador PARTITION en la consulta CREATE TABLE.
En el siguiente ejemplo se crean cuatro particiones por rango de fechas:

```
CREATE TABLE actuaciones (id INT, descripcion VARCHAR(50), fecha DATE)
    ENGINE=INNODB
    PARTITION BY RANGE( YEAR(fecha) ) (
        PARTITION act09 VALUES LESS THAN (2009),
        PARTITION act10 VALUES LESS THAN (2010),
        PARTITION act11 VALUES LESS THAN (2011),
        PARTITION act12 VALUES LESS THAN (2012)
    );
```

Ejemplo de como crear un particionado por Lista, donde cada particionado contiene dos años:

```
CREATE TABLE actuaciones (id INT, descripcion VARCHAR(50), fecha DATE)
    ENGINE=INNODB
    PARTITION BY LIST(YEAR(fecha)) (
        PARTITION act01 VALUES IN (2009, 2010),
        PARTITION act02 VALUES IN (2011, 2012)
    );
```

Ejemplo de como crear un particionado por *Hash*, donde se pasan la clave utilizada para el *Hash* y la cantidad de particiones elegida:

```
CREATE TABLE actuaciones (id INT, descripcion VARCHAR(50), fecha DATE)
    ENGINE=INNODB
    PARTITION BY HASH( YEAR(fecha) )
    PARTITIONS 6;
```

Ejemplo de como crear un particionado por KEY, donde se pasa únicamente la cantidad de particiones deseada:

```
CREATE TABLE actuaciones (id INT, descripcion VARCHAR(50), fecha DATE)
    ENGINE=INNODB
    PARTITION BY KEY()
    PARTITIONS 6;
```

Cuando la finalidad del particionamiento es optimizar el acceso a los datos, también se puede acompañar con la utilización de varios discos físicos con lo cual la carga de operaciones E/S será distribuida. Para ello MySQL proporciona la capacidad especificar en qué directorio se desea guardar la tabla particionada como lo muestra el siguiente ejemplo:

```
CREATE TABLE actuaciones (id INT, descripcion VARCHAR(50), fecha DATE)
    ENGINE=INNODB
    PARTITION BY KEY() (
        PARTITION p0 DATA DIRECTORY='/data/mysql/act_table_p0/',
        PARTITION p1 DATA DIRECTORY='/data/mysql/act_table_p1/'
    );
```

Con la opción "DATA DIRECTORY", MySQL puede definir en que directorio se almacenará la partición de la tabla.

3.4.3.3. Limitaciones del particionado de tablas

- El motor de almacenamiento debe ser el mismo para todas las particiones de una misma tabla
- Los motores de almacenamiento MERGE, CSV o FEDERATED no permiten el uso del particionamiento
- El particionado por clave solo está admitido si se utiliza como motor de almacenamiento NDBCLUSTER
- No se admite el uso de funciones, *stored procedures* o *plugins*, ni la declaración de variables en las expresiones de particionamiento
- Pueden utilizarse operadores aritméticos o lógicos solamente si devuelven un resultado de tipo entero o NULL. Se exceptúa de este caso el particionado por clave
- No se admite el uso de operadores a nivel de bits
- El tamaño máximo de particiones y subparticiones para una tabla es de 1024
- El particionamiento no soporta claves foráneas, índices *FULLTEXT* ni tipos de datos *POINT* o *GEOMETRY*. Tampoco existe soporte para la caché de claves ni para la opción *DELAYED*
- No pueden particionarse ni tablas temporales ni tablas de logs
- Una clave de particionado no puede ser una subconsulta
- mysqlcheck y myisamchk no pueden aplicarse a tablas particionadas

3.4.4. Particionado en PostgreSQL

El particionado en PostgreSQL[27], se basa principalmente en la funcionalidad de Herencia. Básicamente a nivel de PostgreSQL se define la herencia como un mecanismo en el cual se utiliza la definición de una tabla la cual denominaremos "Padre", para definir una nueva tabla llamada "Hija" que puede heredar sus columnas y demás funcionalidades.

Las tablas pueden heredar de mas de una tabla "Padre" y contener mas de una tabla "Hija". PostgresSQL, implementa dos tipos de particionado, por Rangos y por Listas los cuales son definidos en la declaración de la tabla.

3.4.4.1. Qué se necesita para utilizar el particionado de tablas con PostgreSQL

Para que las consultas que trabajan sobre tablas particionadas funcionen correctamente se debe cambiar la configuración de la base de datos y modificar el parámetro de opciones de planeamiento "constraint_exclusion" el cual por defecto se encuentra deshabilitado y se debe cambiar por "partition". Este parámetro se encuentra en el archivo postgresql.conf.

Con la opción habilitada, los datos de la tabla se organizan en particiones, donde cada partición impone restricciones (constraints) distintas a su subconjunto de datos. De esta forma, un registro se ubica en una partición junto a otros registros que cumplen con las

restricciones. El optimizador puede calcular de antemano en qué partición se encuentra un registro, disminuyendo el conjunto de búsqueda. La desventaja es que para cada consulta se debe procesar las restricciones para localizar las particiones que contienen los datos objetivo.

3.4.4.2. Creación de particiones

La creación de tablas particionadas consta de una serie de pasos descriptos a continuación. Definir la tabla padre, la cual servirá de referencia y para que las tablas hijas hereden sus características:

```
CREATE TABLE actuaciones (
id integer NOT NULL,
descripcion VARCHAR(50),
fecha DATE
);

ALTER TABLE actuaciones OWNER TO postgres;
```

Se crean las tablas que heredarán de la tabla “actuaciones”:

```
CREATE TABLE act2010(
CONSTRAINT chk_fecha_2010 CHECK (fecha >= '2010-01-01'::date AND fecha <= '2010-12-31'::date)
)
INHERITS (actuaciones);
ALTER TABLE act2010 OWNER TO postgres;

CREATE TABLE act2011(
CONSTRAINT chk_fecha_2011 CHECK (fecha >= '2011-01-01'::date AND fecha <= '2011-12-31'::date)
)
INHERITS (actuaciones);
ALTER TABLE act2011 OWNER TO postgres;

CREATE TABLE act2012(
CONSTRAINT chk_fecha_2012 CHECK (fecha >= '2012-01-01'::date AND fecha <= '2012-12-31'::date)
)
INHERITS (actuaciones);
ALTER TABLE act2011 OWNER TO postgres;
```

Con la estructura de las tablas definida es necesario crear el direccionamiento de la tabla “padre” a las tablas “hijas”

```
CREATE OR REPLACE FUNCTION act_insert_trigger()
  RETURNS trigger AS
$BODY$
begin
if (new.fecha >= date '2010-01-01' and new.fecha<= date '2010-12-31') then
    insert into act2010 values (new.*);
elsif (new.fecha >= date '2011-01-01' and new.fecha <= date '2011-12-31') then
    insert into act2011 values (new.*);
elsif (new.fecha >= date '2012-01-01' and new.fecha <= date '2012-12-31') then
    insert into act2012 values (new.*);
else
    raise exception 'Rango de fecha desbordado, amplie la funcion en el trigger act_insert_trigger()';
end if;
return null;
end;
$BODY$
LANGUAGE 'plpgsql' VOLATILE
COST 100;
ALTER FUNCTION act_insert_trigger() OWNER TO postgres;
```

Se debe crear un *trigger* que intercepte el ingreso de registros y que ejecute el código anterior

```
CREATE TRIGGER insert_act_trigger
  BEFORE INSERT
  ON actuaciones
  FOR EACH ROW
  EXECUTE PROCEDURE act_insert_trigger();
```

3.5. Concurrencia

A los efectos de permitir el uso concurrente de la base de datos y a su vez mantener la consistencia de los datos es necesario que el RDMBS utilice algún tipo de mecanismo que la garantice. A este mecanismo -en el contexto de Base de Datos- se le llama Bloqueo o por su denominación en inglés *Lock*. Es la forma en que la transacción que está realizando una acción sobre la base de datos se asegura que los recursos involucrados no puedan ser alterados por otras transacciones. Existen diversos tipos de implementación y niveles de reserva de los recursos dependiendo de cada RDBMS.

En general los RDBMS utilizan dos tipos de bloqueo llamados "bloqueo de lectura" (o bloqueo compartido) y "bloqueo de escritura" (o bloqueo exclusivo), el primero es utilizado cuando se realizan consultas del recurso -por lo que puede ser compartido con otras transacciones que solo realizan consultas- pero no con transacciones que intentan modificar el contenido y el segundo es utilizado cuando una transacción desea modificar el contenido impidiendo el acceso al recurso de otra transacción.

La utilización de bloqueos puede llevar a un estado denominado *DeadLock*, el cual se describirá con un ejemplo. Supóngase que la transacción T1 bloquea al recurso R1 y la transacción T2 bloquea el recurso R2. Ahora la transacción T1 continúa con su trabajo y se dispone a modificar el recurso R2, pero se encuentra que está bloqueado por T2. A su vez la transacción T2 se dispone a proseguir su trabajo pero necesita modificar el recurso R2 el cual se encuentra bloqueado por T1. De esta manera las dos transacciones quedan esperando indefinidamente *DeadLock* a que la otra libere el recurso para tomar control del mismo.

Cuando se habla de bloqueos y dado que las bases de datos candidatas deben cumplir con las propiedades ACID, no se puede dejar de mencionar la propiedad de Aislamiento o *Isolation* que las transacciones poseen.

El estándar ANSI/ISO SQL define cuatro niveles de aislamiento transaccional:

- **Lectura Secuenciable** "*SERIALIZABLE*" es el nivel más alto de aislamiento. Para garantizar una lectura de este tipo, las transacciones no deben liberar sus bloqueos hasta finalizar. Además, una consulta del tipo SELECT WHERE, debe adquirir un bloqueo sobre todo el rango de datos, evitando las lecturas fantasmas.

- **Lectura Repetible** "*REPEATABLE READ*" es el nivel de aislamiento en el que las transacciones no liberan bloqueos hasta su fin, pero no se implementan bloqueos de rango, por lo que pueden ocurrir lecturas fantasmas.

- **Lectura comprometida** "*READ COMMITTED*" exige que los bloqueos de escritura no sean liberados por la transacción hasta su fin. Pero, los bloqueos de lectura

son liberados inmediatamente que la operación SELECT finaliza, permitiendo lecturas no repetibles.

- **Lectura no comprometida** "*READ UNCOMMITTED*" -el nivel más débil de aislamiento-, las consultas retornan datos que pueden haber sido alterados por otras transacciones, pero no confirmados, dando lugar lecturas sucias.

Estos niveles de aislamiento son definidos en base a la ocurrencia de una serie de eventos, los cuales a continuación son detallados. La tabla 3.1 resume qué eventos ocurren en cada nivel de aislamiento.

- **Lectura sucia**. Las sentencias SELECT son ejecutadas sin realizar bloqueos, pero podría usarse una versión anterior de un registro. Por lo tanto, las lecturas no son consistentes al usar este nivel de aislamiento.
- **Lectura no repetible**. Una transacción vuelve a leer datos que previamente había leído y encuentra que han sido modificados o eliminados por una transacción cursada.
- **Lectura fantasma**. Una transacción vuelve a ejecutar una consulta, devolviendo un conjunto de registros que satisfacen una condición de búsqueda y encuentra que otros registros que satisfacen la condición han sido insertadas por otra transacción cursada.

A modo de resumen, el cuadro 3.1 describe que eventos ocurren en cada nivel de aislamiento.

Nivel de Aislamiento	**Sucia**	**No repetible**	**Fantasma**
Secuenciable	No	No	No
Lectura Repetible	No	No	Sí
Lectura Comprometida	No	Sí	Sí
Lectura No Comprometida	Sí	Sí	Sí

Cuadro 3.1: Niveles de Aislamiento definidos en el Estándar ANSI/ISO SQL

Haciendo una comparación entre los niveles de aislamiento y los tipos de bloqueos, el cuadro 3.2

Nivel de Aislamiento	**Bloqueo Escritura**	**Bloqueo Lectura**
Secuenciable	Sí	Sí
Lectura Repetible	Sí	Sí
Lectura Comprometida	Sí	No
Lectura No Comprometida	No	No

Cuadro 3.2: Niveles de Aislamiento definidos por Tipos de Bloqueos utilizados

A mayor grado de aislamiento, mayor precisión, pero a costa de menor concurrencia.

La técnica MVCC (*Multi Version Concurrency Control*) tiene como principal característica que realiza una copia del recurso utilizado. De esta forma no es necesario un bloqueo para mantener la consistencia de los datos, es decir, cada transacción tiene su propia versión y no genera conflicto con otra. Si dos transacciones modificaron la misma

tabla, se hace un *merge* de ambas tablas combinando las últimas versiones de cada registro. Si las dos transacciones modificaron exactamente el mismo registro, entonces en ese caso, cuando se realice el *commit*, el registro que finalmente prevalece, será el resultado de la operación de la última transacción sobre ese registro. Es muy habitual usar este tipo de técnica de concurrencia para evitar los *deadlocks*, sin embargo, MVCC no debe ser usado en cualquier ocasión, ya que tiene un *overhead* muy importante debido a que las versiones de registros se copian y almacenan en tablas o estructuras temporales que luego deben eliminarse. La implementación interna de este algoritmo es distinta en cada DBMS.

En esta sección se expondrá la forma en que cada DBMS implementa técnicas de concurrencia.

3.5.1. Concurrencia en MySQL

InnoDB permite utilizar los cuatro niveles de aislamiento definidos en el estándar ANSI/ISO SQL. El nivel deseado de aislamiento se establece configurando el archivo "my.cfg" o a través de la siguiente sentencia[18]:

```
SET [SESSION | GLOBAL] TRANSACTION ISOLATION LEVEL
   {READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE}''
```

Implementa el control de bloqueos a través del protocolo *Bloqueo de Granularidad Múltiple*[6] (Multiple Granularity Locking), por el cual además de los bloqueos exclusivo (X) y compartidos (S), existen otros dos tipos de bloqueos denominados "intencional exclusivo" (IX) e "intencional compartido" (IS).

Los bloqueos X y S son bloqueos a nivel de fila mientras que los bloqueos IX e IS son a nivel de tabla, la principal funcionalidad de estos últimos -como su nombre lo indica- es establecer la intención que tiene una transacción de bloquear una fila de la tabla. De esta forma, un X o S es precedido por un IX o IS respectivamente.

A su vez también utiliza la implementación de MVCC denominada en InnoDB como Consulta Consistente, por la cual una consulta obtiene una imagen del recurso con los cambios confirmados hasta el momento y no los realizados con posterioridad. Si el motor se encuentra en modo *Repeteable Read* o *Read Commited* las sentencias SELECT utilizaran las consultas consistentes por lo que no se generará un bloqueo sobre el recurso.

InnoDB posee un sistema de detección automática de *DeadLock* basado en un grafo de esperas por recursos. Si se genera un ciclo en este grafo, se entra en el estado de *Deadlock*. MySQL elige la transacción que ha modificado menos cantidad de filas como víctima, y la deshace. Otra forma de evitar *Deadlocks* es configurar el parámetro *innodb_lock_wait_timeout* con un valor razonable de tiempo de espera para un recurso. Al finalizar este tiempo, la transacción de deshará.[37]

Caso de Prueba

Entre los casos de prueba realizados contra ambos motores se intentó asegurar que los pedidos y liberaciones de bloqueos se realizaran de manera adecuada. Las consultas siguientes se realizaron en ambos motores:

1.
```
BEGIN
FOR i=1 TO 100000
   INSERT INTO <table> VALUES (...)
END FOR
COMMIT
```

2.
```
BEGIN
  ALTER TABLE <table> ADD COLUMN <new_col> <new_type>
COMMIT
```

El resultado obtenido utilizando el motor InnoDB de MySQL mostró que el *lock* de la tabla para escritura fue correctamente obtenido por la primera transacción y la inserción comenzó correctamente. Mientras se estaba ejecutando la primera transacción fue lanzada la segunda, intentando modificar la estructura de la tabla sobre la que se estaban insertando datos. Sin embargo, al lanzar la segunda transacción a la vez quedó a la espera del *lock* de escritura, se produjo, como efecto inesperado la detención de la inserción masiva de datos lanzada por la primera consulta.

3.5.2. Concurrencia en PostgreSQL

PostgreSQL implementa el modelo MVCC desde la versión 8.3. por el cual las transacciones ven una imagen de los recursos al momento de utilizar los datos (para ello los recursos se versionan con un *timestamp*), esto impide que se generen inconsistencias sobre los datos, cuando otras transacciones realizan operaciones de lectura y escritura estos mismos.[26]

Además PostgreSQL también provee la utilidad de bloquear un recurso en forma explícita tanto a nivel de tabla como a nivel de fila.

En simple, el MVCC nunca modifica o elimina la datos sino que nuevas filas de información se van añadiendo conforme se crea o actualizan los datos y se marca la versión anterior de los datos como "no visible". Cuando se desea eliminar un dato igualmente se añade una fila y se marca como "no visible" al mismo tiempo. Los datos nunca son "visibles" por otros usuarios hasta que no sea confirmada (COMMIT). La principal ventaja es que las operaciones de R nunca bloquean a las de W, y viceversa, podemos obtener respaldos en caliente sin bloquear la base de datos. Como desventaja: consumimos mas disco duro.

Caso de Prueba

A diferencia de lo sucedido bajo InnoDB, con PostgreSQL el funcionamiento de lanzar una consulta ALTER TABLE mientras corría una transacción insertando datos masivamente, tuvo un comportamiento según lo esperado.

En primer lugar la transacción obtiene los bloqueos sobre las filas que va insertando. Cuando se dispara la consulta "ALTER TABLE", esta pide un bloqueo exclusivo sobre toda la tabla. Dado que es imposible satisfacerlo debido a los bloqueos parciales obtenidos por la transacción en curso, la consulta "ALTER TABLE" queda esperando que se le otorgue el bloqueo pedido.

La figura 3.3 muestra el pedido del bloqueo exclusivo por parte de la consulta "ALTER TABLE" que no será otorgado hasta finalizar la transacción.

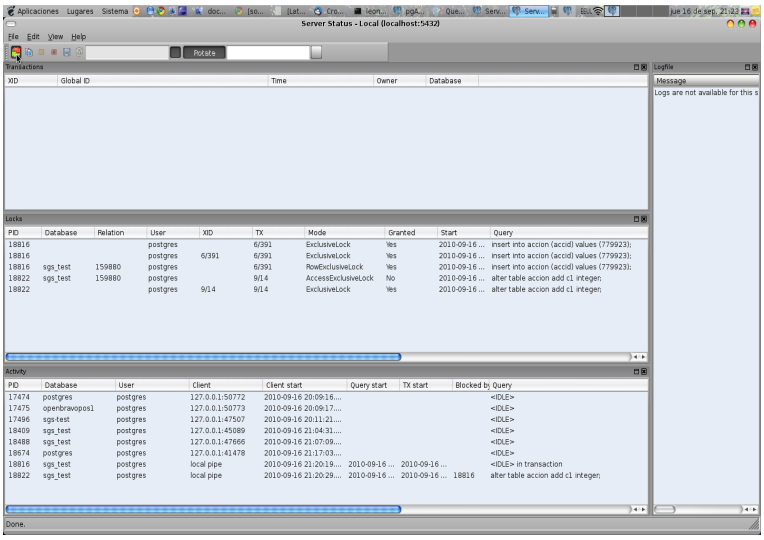

Figura 3.3: Vista de bloqueos en PostgreSQL

3.5.3. El caso "kill -9" en PostgreSQL

El caso surge al intentar matar un proceso corriendo sobre la base de datos, haciendo uso de la señal SIGKILL (kill -9) en lugar de utilizar la señal SIGTERM (kill -15). Dado que cada proceso corre en un hilo separado y que existe un proceso para el servidor PostgreSQL, no sería esperable que al matar a uno de los procesos, los otros se vieran afectados.

Sin embargo, al enviar la señal SIGKILL a cualquiera de estos procesos, todos los demás abortan. Según los registros de PostgreSQL, el problema surge al no poder asegurar que la memoria compartida no esté corrupta.[31]

Si bien podría sostenerse que el motor de la base de datos no debiera estar obligado a responder adecuadamente ante una señal SIGKILL, sería deseable un comportamiento similar al de otros motores que no presentan este problema.

No se ha detectado que se presenten problemas al utilizar la señal SIGTERM

3.6. Optimización de Consultas

Para cada consulta realizada sobre el DBMS, existe más de una opción en la estrategia para recolectar los datos. Se puede recorrer toda la tabla o utilizar índices disponibles, en el caso de JOINs, se pueden utilizar diferentes métodos y ordenamientos. La combinación de decisiones tomadas en cuando a qué método utilizar en cada paso hasta resolver la consulta crea un grafo de pasos dependientes, representado habitualmente como un árbol de operaciones.

Ejemplo: MERGE JOIN sobre dos tablas

Operaciones:

- ordenamiento sobre tabla 1

- ordenamiento sobre tabla 2
- merge del resultado de 1. y 2.

La raíz del árbol de ejecución representa al resultado final de la consulta.

El optimizador del DBMS compara todos árboles de ejecución posibles y determina el costo de cada uno en términos de cantidad de operaciones a disco, ciclos de CPU y consumo de red. A cada recurso se le asigna un intervalo de tiempo dado de manera que se pueda elegir el árbol con menos costo[32]. El optimizador podrá elegir en base al mejor tiempo general o al mejor tiempo de respuesta.

En el caso de las bases de datos, la interacción con el disco cobra mayor importancia debido a que el sistema mecánico para posicionar la cabeza lectora sobre los datos es mucho más lento que operaciones de la CPU sobre memoria, y elegir un plan de ejecución equivocado puede resultar en una pérdida muy significativa de rendimiento.

Resulta evidente que para consultas complejas no es posible considerar todos las combinaciones posibles ya que se emplearía demasiado tiempo en decidir cuál es la mejor. El optimizador se ve forzado entonces en dejar de evaluar algunas combinaciones aumentando la probabilidad de que el árbol elegido no sea el óptimo en realidad.

Los costos de cada operación se estiman en base a estadísticas de tales como la cantidad de filas que tiene una tabla y la dispersión de valores en índices. De esta forma, el optmizador puede equivocarse al elegir si las estadísticas están desactualizadas, especialmente luego de una inserción o eliminación de gran cantidad de datos.

Cuando el DBA detecta que el optimizador no está eligiendo un plan de ejecución óptimo debe aplicar las técnicas de optimización disponibles según el sistema de base de datos utilizado.

Los RDBMS tradicionales cuentan con operaciones específicas para recalcular las estadísticas de la base de datos en caso de que éstas estén impidiendo que el optimizador infiera un plan óptimo. Normalmente estas operaciones requieren un uso del sistema muy intenso y un largo tiempo de ejecución por lo que no siempre pueden ser ejecutadas a demanda. En este escenario, es necesario contar con técnicas que permitan al DBA mejorar el rendimiento de consultas específicas. Se detallarán dos de ellas: utilización de "*hints*" y alteración de estadísticas.

Hints

El encargado de resolver una consulta SQL es el propio DBMS. Los optimizadores modernos son del tipo basado en costos, como se describió más arriba y son muy sensibles a la exactitud de las estadísticas de la base de datos. Los *hints* son mecanismos por los cuales el programador de la consulta puede sugerirle al optimizador cómo elegir el plan de ejecución.

Al encontrarse un escenario en el que el optimizador trabaja en forma incorrecta, la solución debería ser solucionar el problema, ya sea mejorando las estadísticas o alterando el esquema de la base de datos (esto último no siempre es posible en sistemas en producción). Se ha sugerido que utilizar *hints* es una práctica pobre por parte del DBA debido a incapacidad de detectar y solucionar el problema utilizando herramientas y técnicas

modernas[5] y forman parte del repertorio antiguo de técnicas de optimización cuando los DBMS utilizaban optimizadores basados en reglas[III].

De cualquier manera en la práctica podrían existir casos en los que se necesitara optimizar una consulta particular en forma urgente. Los *hints* podrían ser útiles en esas instancias.

Alteración de estadísticas

Algunos DBMS proveen mecanismos para alterar las estadísticas de la base de datos puntualmente para forzar al optimizador a elegir un plan de ejecución diferente. Se aplicará esta técnica a PostrgreSQL ya que MySQL no provee un mecanismo de acceso a las estadísticas que no sea recrearlas totalmente, por medio de la sentencias ANALYZE TABLE y OPTIMIZE TABLE.

Se detallará a continuación el uso de *hints* en MySQL y de alteración de estadísticas en PosgreSQL.

3.6.1. MySQL

En MySQL, es posible darle al optmizador indicaciones o *hints* sobre qué indices elegir durante el procesamiento de consultas. Estas indicaciones se escriben como parte de la sentencia SQL y van inmediatamente después de la referencia a la tabla, siendo posible utilizar cualquiera de estas tres: “IGNORE INDEX”, “USE INDEX” y “FORCE INDEX”.

USE INDEX y FORCE INDEX permiten indicar una lista de índices para que el optimizador considere seleccionar los datos utilizando los índices dados. La característica particular de FORCE INDEX es que además le indica al optimizador que hacer un recorrido completo de la tabla sería muy costoso. Recordar que el optimizador podría decidir no utilizar un índice si este contuviera relativamente pocos valores del total de la tabla. IGNORE INDEX le sugiere al optimizador no tomar el índice en cuenta al decidir.

Ejemplo: Dado el siguiente esquema:

```
CREATE TABLE ‘lecturas‘ (
  ‘id‘ int(11) NOT NULL AUTO_INCREMENT,
  ‘dispositivo_id‘ int(11) NOT NULL DEFAULT ’0’,
  ‘creado_el‘ TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  ‘datos‘ varchar(25) DEFAULT NULL,
  ‘dato1‘ varchar(255) DEFAULT NULL,
  ‘dato2‘ text DEFAULT NULL,
  ‘dato3‘ BLOB DEFAULT NULL,
  ‘dato4‘ longblob DEFAULT NULL,
  ‘dato5‘ longtext DEFAULT NULL,
  ‘dato6‘ char(255) DEFAULT NULL,
  ‘dato7‘ char(255) DEFAULT NULL,
  PRIMARY KEY (‘id‘),
  KEY ‘lecturas_dispositivo_id_creado_el‘ (‘dispositivo_id‘,‘creado_el‘)
) ENGINE=InnoDB AUTO_INCREMENT=174563 DEFAULT CHARSET=latin1;
```

Luego de una carga de datos extensa, se llega a la siguiente situación:

[III]Los RBO (*Rule Based Optimizer*) trabajan fundamentalmente utilizando reglas según la cláusula WHERE. Por ejemplo, si existe un índice sobre una columna, siempre se utilizará independientemente de su cardinalidad. Utilizar un índice cuando la cantidad de valores devueltos supera cierto porcentaje de la cardinalidad de la tabla[40], el rendimiento se ve reducido

```
mysql> explain select count(*) from lecturas where id between 28865510 and
40100000 and dispositivo_id > 2;
+----+-------------+-----------+-------+-----------------------------------------+
| id | select_type | table     | type  | possible_keys                           |
+----+-------------+-----------+-------+-----------------------------------------+
|  1 | SIMPLE      | lecturas  | range | PRIMARY,lecturas_dispositivo_id_creado_el |
+----+-------------+-----------+-------+-----------------------------------------+

-----------------------------------+---------+------+----------+-------------------------+
 key                               | key_len | ref  | rows     | Extra                   |
-----------------------------------+---------+------+----------+-------------------------+
 lecturas_dispositivo_id_creado_el | 4       | NULL | 24000274 | Using where; Using index |
-----------------------------------+---------+------+----------+-------------------------+

1 row in set (0.16 sec)
```

La sentencia explain muestra que las posibles claves son PRIMARY y "lecturas_dispositivo_id_creado_el". "lecturas_dispositivo_id_creado_el" será la clave elegida por el optimizador.

Si se ejecuta la consulta:

```
mysql> select count(*) from lecturas where id between 28865510 and 40100000 and
dispositivo_id > 2;
+-----------+
| count(*) |
+-----------+
|   9371007 |
+-----------+
1 row in set (2 min 5.77 sec)
```

Se observa que el tiempo de ejecución de la consulta es de poco más de dos minutos, habiendo el optimizador optado por utilizar la clave "lecturas_dispositivo_id_creado_el". Si se ejecuta esta consulta, ahora utilizando un *hint*:

```
mysql> select count(id) from lecturas force index (PRIMARY)
where id between 28865510 and 40100000 and dispositivo_id > 2;
+-----------+
| count(*) |
+-----------+
|   9371007 |
+-----------+
1 row in set (13.11 sec)
```

Ejecutando la misma consulta, pero sugiriéndole a MySQL que fuerce la clave PRIMARY, el mismo resultado se devuelve en menos de 15 segundos.
Las cantidades reales de la tabla son:

```
mysql> select count(*) from lecturas;
+-----------+
| count(*) |
+-----------+
|  48000000 |
+-----------+
1 row in set (2 min 0.68 sec)

mysql> select count(id) from lecturas where id between 28865510 and 40100000;
+-----------+
| count(*) |
+-----------+
|  11234491 |
+-----------+
1 row in set (15.95 sec)

mysql> select count(id) from lecturas where dispositivo_id > 2;
```

```
+-----------+
| count(*) |
+-----------+
|  43697494 |
+-----------+
1 row in set (1 min 42.08 sec)
```

El optimizador eligió el plan en forma incorrecta, ya que el segundo índice trae inicialmente un conjunto de datos casi cuatro veces mayor que el primero, aunque ya lo hace en forma ordenada. Lo cierto es que recuperar el conjunto menor y ordenarlo por "creado_el" resultó más eficiente, hecho que el optimizador no anticipó.

Antes de ejecutar cada consulta se reinició el servidor de MySQL para evitar que fueran afectadas por el caché de consultas. La prueba también se realizó utilizando SQL_NO_CACHE dando los mismos resultados.

Como se ha comentado más arriba, las estadísticas de una tabla pueden estar desactualizadas causando una importante pérdida de rendimiento. Existen en MySQL dos comandos para solucionar este problema: ANALYZE TABLE y OPTIMIZE TABLE.

ANALYZE TABLE

Este comando analiza y almacena la distribución de claves para una tabla. Durante el análisis, la tabla se bloquea con un bloqueo de lectura. ANALYZE TABLE retorna un conjunto de resultados con los siguientes valores:

El uso de ANALYZE TABLE en producción ha de usarse con precaución ya que se ha sugerido que bajo ciertas condiciones puede llevar a un bloqueo no desado de la base de datos[41]

OPTIMIZE TABLE

Según la documentación de MySQL[14], este comando debería ser ejecutado si se ha eliminado una gran parte de la tabla o si se ha hecho muchos cambios a una tabla con columnas de ancho variable (VARCHAR, VARBINARY, BLOB o TEXT). OPTIMIZE TABLE se utiliza para restaurar el espacio libre en disco y defragmentar el archivo de datos.

```
mysql> optimize table lecturas;
+-----------------+---------+---------+-------------------------------------------------------------------+
| Table           | Op      | Msg_type| Msg_text                                                          |
+-----------------+---------+---------+-------------------------------------------------------------------+
| pg_test.lecturas| optimize| note    | Table does not support optimize, doing recreate + analyze instead|
| pg_test.lecturas| optimize| status  | OK                                                                |
+-----------------+---------+---------+-------------------------------------------------------------------+
2 rows in set (41 min 14.63 sec)
```

MySQL bloquea la tabla mientras este comando se ejecuta y consume un tiempo importante de ejecución.

```
mysql> show processlist;
+-----+------+-----------+---------+---------+------+---------------------------------+
| Id  | User | Host      | db      | Command | Time | State                           |
+-----+------+-----------+---------+---------+------+---------------------------------+
| 376 | root | localhost | pg_test | Query   | 1303 | copy to tmp table               |
| 377 | root | localhost | pg_test | Query   |    0 | NULL                            |
| 378 | root | localhost | pg_test | Query   | 1240 | Waiting for table metadata lock |
+-----+------+-----------+---------+---------+------+---------------------------------+
-------------------------------------------------+
 Info                                            |
-------------------------------------------------+
 optimize table lecturas                         |
```

```
 show processlist                                  |
 insert into lecturas  (dispositivo_id) values (200) |
-----------------------------------------------------+
3 rows in set (0.78 sec)
```

El motor InnoDB se beneficia de la ejecución de este comando, pero debe ser utilizado con cuidado en producción debido a que reconstruye la tabla e impide el acceso de otras consultas a ella durante su ejecución. Habitualmente es necesario buscar otras opciones más "livianas" para regenerar la tabla[42]

3.6.2. PostgreSQL

Las estadísticas utilizadas para la planificación de ejecución de consultas son almacenadas en el catálogo de sistema "pg_statistic", cuyas entradas son actualizadas por ANALYZE y VACUUM ANALYZE, y siempre son estimadas, aun cuando se hayan recolectado recientemente.

La cantidad de información almacenada por ANALYZE puede establecerse de dos maneras:

1. columna por columna usando el comando ALTER TABLE SET STATISTICS

2. globalmente ajustando "default_statistics_target"

El límite por defecto es de 1000 entradas para la versión 8.4. Subiendo este límite permitiría hacer estimados mas precisos, particularmente para columnas con una distribución de datos irregular, al precio de consumir más espacio en "pg_statisticz ligeramente un poco más de tiempo para computar los estimados.

La configuración debe ser habilitada desde el archivo de configuración "postgres.conf".

El parámetro **track_counts** controla si se recopilan las estadísticas de las tablas y los accesos a índice.

El parámetro **track_functions** permite el seguimiento del uso de funciones definidas por el usuario.

El **track_activities** permite el control del comando actual que está siendo ejecutado por cualquier proceso de servidor.

3.6.2.1. La vista pg_stats

Se muestra la estructura de la vista que recolecta las estadísticas con el detalle de los campos involucrados:

Nombre	Tipo	Referencias	Descripción
schemaname	name	pg_namespace.nspname	Nombre del esquema
tablename	name	pg_class.relname	Nombre de la tabla
attname	name	pg_attribute.attname	Nombre de columna descripta por esta fila
null_frac	real		Fracción de entradas de la columna que son nulas
avg_width	integer		Ancho promedio, en bytes, de las entradas de la columna
n_distinct	real		Si es mayor que cero, el número estimado de valores distintos en la columna. Si es menor que cero, es el negativo del número de valores distintos dividido entre el número de filas (La forma negativa se usa cuando ANALYZE cree que el número de valores diferentes tiende a crecer a medida que crece la tabla; la forma positiva se usa cuando la columna parece tener un número fijo de valores posibles Por ejemplo, -1 indica una única columna en la que el número de valores distintos es el mismo que el número de filas
most_common_vals	anyarray		Lista de los valores más comunes en la columna. (NULL si no hay valores que sean más comunes que otros.) Para algunos tipos de datos, como tsvector, esto es una lista de los valores más comunes de los elementos más que los valores del tipo mismo.
most_common_freqs	real[]		Lista de las frecuencias de los valores o elementos más comunes, p.ej. el número de ocurrencias de cada uno dividido entre el total de filas (NULL cuando most_common_vals también lo es.) Para algunos tipos de datos como tsvector, puede contener información adicional, haciéndolo mas largo que el array most_common_vals array.
histogram_bounds	anyarray		Una lista de valores que divide a los valores de comunes la columna en grupos de población similar. Los valores en most_common_vals, si es que existen, son omitidos en el cálculo de este histograma. (Esta columna es NULL si el tipo de datos de la columna no tiene un operador < o si la lista most_common_vals incluye a toda la población.)
correlation	real		Correlación estadística entre el orden físico de las filas y el orden lógico de los valores de las columnas. El rango de valores es de +1 a -1. Cuando el valor está cercano a +1 o -1, se estima que el escaneo indexado de la columna será más eficiente que cuando el valor se acerca a 0 debido a la reducción de los accesos aleatorios a disco (Esta columna será NULL si el tipo de datos de la columna no tiene un operador <.)

3.6.3. Alteración de estadísticas

Como se ha visto, modificar las estadísticas puede implicar una sustancial alteración del funcionamiento normal de la planeación de ejecución de una consulta.[29] En un join simple de la forma:

```
SELECT * FROM a, b, c WHERE a.id = b.id AND b.ref = c.id;
```

El optimizador tiene la libertad de asociar las tablas en cualquier orden. Podría asociar a la tabla “a” con la tabla “b” y luego asociar al resultado con la tabla “c” o comenzar asociando a “b” con “c” para, finalmente, asociar a la tabla “a” con el resultado obtenido anteriormente.

El punto aquí es que si bien semánticamente las soluciones conducen a resultados equivalentes, podrían tener grandes diferencias en costo de ejecución. El optimizador deberá explorar todas las posibilidades para tratar de encontrar el plan más eficiente.

Sin embargo, tampoco es posible explorar todas las posibilidades cuando crece el número de tablas, debido a que las asociaciones posibles crecen exponencialmente con el crecimiento del número de tablas. La exploración, cuando no corresponda que sea exhaustiva, deberá basarse en probabilidades y quizás no encuentre el mejor de los planes posibles.

Aunque muchos tipos de JOIN no restringen completamente el orden en que se realizan, es posible indicarle al optimizador de PostgreSQL que trate a todas las sentencias JOIN como si, aun así, restringieran el orden. Por ejemplo, estas consultas son lógicamente equivalentes:

```
SELECT * FROM a, b, c WHERE a.id = b.id AND b.ref = c.id;

SELECT * FROM a CROSS JOIN b CROSS JOIN c WHERE a.id = b.id AND b.ref = c.id;

SELECT * FROM a JOIN (b JOIN c ON (b.ref = c.id)) ON (a.id = b.id);
```

Pero si le indica al optimizador que respete el orden de los JOINs, la segunda y la tercera, tomarán menos tiempo en planificarse que la primera. Este efecto no es preocupante para dos o tres tablas, pero puede ser un salvavidas si son muchas.

Para forzar al optimizador a seguir el orden establecido por los JOINs explícitos, al parámetro de ejecución "join_collapse_limit" debe asignársele el valor 1. Para disminuir el tiempo de búsqueda, no es necesario restringir completamente el orden de los "JOINs". En la consulta:

```
SELECT * FROM a CROSS JOIN b, c, d, e WHERE ...;
```

si el valor de join_collapse_limit es 1, se fuerza al optimizador a unir a las tablas A y B antes de hacer el "JOIN" con las otras tablas, pero no establece ninguna otra restricción. En el ejemplo, el número de órdenes de "joins" se reduce en un factor de 5.

3.6.4. Hacking de estadísticas en PostgreSQL

Esta sección busca responder al problema que se plantea ante una mala decisión del optimizador. Dado que las decisiones del optimizador en PostgreSQL están basadas en los datos de la vista pg_stats que, a su vez, toma sus valores de la tabla pg_statistic, si se pudieran alterar los valores de esta tabla se estaría cambiando la información de la que dispone el optimizador.

Eventualmente, "falsear" la información acerca de la distribución o de la frecuencia de los valores, puede ser una forma de corregir una mala opción del optimizador influyendo de manera directa en la decisión final. A continuación se muestra un sencillo ejemplo en el que alterando las estadísticas, el optimizador cambia el plan de ejecución de una consulta.

Se considerarán tres campos de la vista pg_stats, a saber: most_common_vals, histogram_bounds y most_common_freqs. El primero de ellos mantiene una estadística de los valores que más se repiten, el segundo los límites considerados para el histograma y el tercero la frecuencia para cada rango del histograma. Los valores correspondientes a dichos campos en la tabla pg_statistic aparecen en los campos stavalues1 para most_common_vals, stanumbers1 para most_common_freqs y stavalues2 para histogram_bounds.

A los efectos de realizar la prueba se creó una tabla sencilla y se alteraron directamente los valores de los campos antedichos en la tabla pg_statistic observando el comportamiento del optimizador antes y después de los cambios.
La tabla se creó según la siguiente consulta SQL:

```
CREATE TABLE teststats
(
   prim_id integer NOT NULL,
   sec_id integer NOT NULL,
   CONSTRAINT idx PRIMARY KEY (prim_id)
)
```

Una vez creada la tabla, se cargó con alrededor de 500000 valores con una distribución variable sobre el campo clave prim_id y tras esto se ejecutó la siguiente consulta SQL:

```
SELECT * FROM teststats WHERE prim_id = 10;
```

Al consultar al analizador por los valores en las estadísticas para la tabla teststats se obtuvieron los siguientes resultados

```
teststats=# select * from pg_stats where tablename = 'teststats' and attname='prim_id';
schemaname | tablename | attname | null_frac | avg_width | n_distinct |  most_common_vals
|------------+-----------+---------+-----------+-----------+------------+--------------------+
 public     | teststats | prim_id |         0 |         4 |  -0.347567 | {270,2301,28398...}|

      most_common_freqs      | histogram_bounds | correlation
-----------------------------+------------------+---------------
 {0.000133333,0.000133333...}| {2,761,1655...}  |    0.666327
(1 row)
```

El resultado indica que los valores más frecuentes encontrado son 27, 2301, etc y que las frecuencias de los intervalos 2-761 y 761-1655 es de alrededor del 0.013 %. La distribución en los valores de prim_id hace que al buscar los resultados con un valor concreto, la opción del optimizador sea utilizar el índice.

```
teststats=# explain select * from teststats where prim_id = 10;

                            QUERY PLAN
 Index Scan using idx on teststats  (cost=0.00..14.38 rows=3 width=8)
   Index Cond: (prim_id = 10)
(2 rows)
```

La modificación realizada sobre la tabla pg_statistic se refleja en la vista siguiente:

```
teststats=# select * from pg_stats where tablename = 'teststats' and attname='prim_id';
 schemaname | tablename | attname | null_frac | avg_width | n_distinct | most_common_vals |
------------+-----------+---------+-----------+-----------+------------+------------------+
 public     | teststats | prim_id |         0 |         4 |  -0.351059 | {10}             |

  most_common_freqs | histogram_bounds | correlation
--------------------+------------------+-------------
 {1}               |                  |    0.662263
(1 row)
```

En ella se observa que aparece como valor más común el valor 10 en un histograma sin límites y con frecuencia 1. Esto indica que los registros cuyo valor en prim_id no es 10 representan una cantidad poco significativa y que, para devolverlos se debe recorrer la tabla no justificando hacer uso del índice.

```
teststats=# explain select * from teststats where prim_id = 10;

                            QUERY PLAN
------------------------------------------------------------------
 Seq Scan on teststats  (cost=0.00..8462.80 rows=499984 width=8)
   Filter: (prim_id = 10)
(2 rows)
```

Por último las estadísticas se reconstruyen al ejecutar la sentencia "vacuum analyze teststats".

3.7. Resumen

Tanto en MySQL como en PostgreSQL es posible realizar todas las tareas relevantes para ASSE utilizando las herramientas disponibles. En lo desarrollado en este capítulo, quizás debiera señalarse en favor de PostgreSQL que el problema detectado con los interbloqueos en MySQL puede generar inconvenientes en el funcionamiento si el administrador de la base de datos lanza una actualización de una tabla mientras se está realizando alguna transacción larga en la base. Por otra parte, la posibilidad de modificar los histogramas y los valores utilizados para la toma de decisiones del planificador también marcan un punto a favor de PostgreSQL.

En el resto de los aspectos, ambos DBMS muestran cumplir adecuadamente con las funcionalidades requeridas.

Capítulo 4

Migración

4.1. Introducción

Este capítulo, tiene por objetivo describir el procedimiento realizado para la puesta en marcha de los RDBMS seleccionados para evaluar su funcionamiento con el sistema SGS. En general la experiencia dicta que la migración de datos, especialmente cuando se trata de muchísimos, presenta un sinnúmero de pequeños problemas que se deben resolver conforme el proceso se va llevando a cabo.

Es necesario planificar la tarea de manera que se vuelva un problema manejable y sobre todo que se pueda ejecutar en etapas o iteraciones, donde se resuelvan problemas e introduzcan mejoras iteración a iteración. En este escenario, se decidió abordar la migración en tres etapas, la primera de ellas la migración de los esquemas de datos en la sección 4.2, luego la etapa de traducción de los *Stored Procedures* y *Triggers* en la sección 4.3 y 4.4 respectivamente y finalmente la migración de datos en la sección 4.5.

4.2. La migración del Esquema de Datos

Como punto de partida en la creación de los esquemas de las bases de datos candidatas, se utilizó la herramienta de generación de Genexus. De los requerimientos, se tiene que las bases de datos MySQL y PostgresSQL son soportadas por Genexus, por lo que a partir de la especificación definida en la KB(*Knowledge Base*) de Genexus y la declaración del conector correspondiente es que se generaron los esquemas para cada base, con sus tipos de datos adecuados. Luego de creados los esquemas de datos de las bases de datos candidatas, se comparó con el esquema de la base de datos Oracle y se vio la necesidad de realizar una serie de ajustes.

A los efectos de un mejor entendimiento, llamaremos "esquemas generados por Genexus" a los esquemas generados a partir de la herramienta Genexus para las bases de datos MySQL y PostgreSQL y "esquema Oracle" al esquema existente en la base de datos Oracle correspondiente al usuario "SSAEPROD" el cual sirve de *back end* a la aplicación SGS.

4.2.1. Diferencias entre el esquema Oracle y los esquemas generados por Genexus

El resultado del estudio comparativo del esquema Oracle con el de las tablas generadas por Genexus marca algunas diferencias que se observan en el cuadro 4.1, las tablas listadas corresponden a tablas ausentes en los nuevos esquemas.

ACT_PROBLEMA	LOCALIDAD4	RESP_PREINS
AUDIT_CONSULTA1530	LOG_NC	RESP_PREINS2
AUDIT_CONSULTA3060	MAX_CENTROS	RESP_PREINSDUP
AUDIT_EJECUTOCONSULTA	MEDFAMILIA	RESP_PREINS_ENV
AUDIT_REPPER	MIRRORACTUACION	RESP_PSGS
AUXFACTURA	MODPROC_JN	RESP_SALESTADO20090403
AUXMUT	MODPROEST_JN	RESP_SALESTADO_200509
BASIMMCALLDUP	MODPROTRA_JN	RESP_SALESTADO_221209
BASIMMCRU1DUP	MONITOREO_PAUTAS	RESP_SERVICIO_030309
BASIMMCRU2DUP	MOVIL_JN	RESP_STFARWF
BASIMMINS	MVO_NUM_PUERTAS	RESP_WF_MAESTRO_CUENTAS
BORRAR_070509	MY_STATS	RESP_WF_MAESTRO_CUENTAS_562
BORRAR_100107	OBJWF_JN	RESQUERY
BORRAR_100107_B	ORDWFRD_JN	RESULTADO1
BORRAR_130309	PANES_PREID	RESULTADO2
BORRAR_140409	PANES_PREID_ENFE	RESUMEN
BORRAR_170809	PANES_PREID_JUL	RESUMEN_SERVICIO
BORRAR_190809	PANES_RUCAF	RESUMEN_SERVICIOCNT
BORRAR_HOY	PANES_RUCAF_DEP	RESUMEN_SEXO
CAMAS_OCUPADAS	PANES_RUCAF_USU	RES_10484819
CENFUN_JN	PANES_USUARIOS	RES_AUX_PRES
CENTOPE_JN	PARAMDEFAULT_JN	RES_OC
CENTPO_JN	PERFSER_JN	RUN_STATS
CENTRO_JN	PERFTRANS_JN	SEGURIDAD_SSAE
CES_CONSDESP	PERSONAL	SEGURIDAD_SSAE_JN
CES_DISTCRO	PERSONAS	SERFUN_JN
CES_DISTDESP	PERSONAS_ORA	SERVICIO_JN
CES_MEDDESP	PERSONAS_PADRON	SSAE_TAB_MOD
CES_PORCEN	PININSID	STATS_HIST
CES_USUARIO	PININSID2	STAT_SQLA1
CODIFICADO	PLAN_TABLE	STAT_SQLA2
CREA_TMP	PP	STFAR
CSTATS$PLAN_TABLE	PREINS_BADDATE	STOCK
DESPACHOS_PRUEBA	PRESENTACION_JN	SUBFAMILIA_JN
DESP_EMEPUE	PRESENTAINS	SUBFAMINV
DESP_GUM	PRESTA_D1	SUSINV
DIGMAL_PASTEUR	PRESTA_DUP	SUSTANCIA_JN
DIRVAL	PRESTA_EDAD	T1
DISEDAD	PRESTA_EDAD2	TABLE_CORRUPT_ROWID
DOBLES	PRESTA_HUERFANOS	TIPOCENTRO_JN
DOMINIO_JN	PRESTA_LOG	TIPOUNIDAD_JN
EMPAQUE_JN	PRESTA_PRU	TRAACCION_JN
ERROR_JOBS	PRESTA_REVIVE	TRANSICION_JN
ESTADO_JN	PRESTA_ROTA	TRAOBJWEST_JN
ESTWFRD_JN	PRESTA_STAT	TRAOBJWF_JN
FAMILIA_JN	PRESTA_STATS	TRAWFRD_JN
FAMINV	PROCEDIMIENTO_JN	ULT_OK
FENOMENO_JN	PROCINS_JN	UNIDAD_JN
FUNCIONARIO_JN	PUESTO_JN	UNIFICA_2008
FUNLIC	QUEST_SCRIPT	UNIFICA_2008_USUARIOS
FUNROL_JN	RANGOETAREO1_JN	USUARIOS_MIDES
GXA0192	RANGOETAREO_JN	USUARIOS_PADRON
GXA0205	RESP_ACTUSUARIO	USUSAINTBOIS
HORAANESTPEREIRA	RESP_AUXCOSTOS	USU_A_MODIFICAR
HORAFUNANESTPEREIRA	RESP_CALLES	USU_BORRADOS
IMM	RESP_CIPAUTA	USU_D1
INSINV	RESP_DIAGNOSTICO	USU_PASTEUR
INSSERV_JN	RESP_INSAT_LATANO	USU_PEREIRA_INS
INSTITUCION_JN	RESP_INSSER	USU_PEREIRA_REPETIDOS
INSTPRECIO	RESP_ODONTOLOGIA	USU_PEREIRA_UPD
INSTVAD_JN	RESP_ORI_PAUTA	VADEMECUMINS_JN
INSUMO_JN	RESP_PRECIOXCANT	VADEMECUM_JN
L1	RESP_PRECIOXCANT_01A08	VISTAOBJWF_JN
LIME	RESP_PREID_ACTUACION	WFRD_JN
LOCALIDAD2	RESP_PREID_PREINS	WF_UNIDAD
LOCALIDAD3		

Cuadro 4.1: Tablas que se encuentran en el modelo Oracle y que se encuentran ausentes en los nuevos esquemas generados por Genexus

Si bien la mayor parte de las tablas son tablas de desarrollo o de auditoría, existen otras diferencias que deberían revisarse previamente a una migración.

4.2.2. Tablas ausentes en el esquema Genexus y Oracle

Con el esquema generado por Genexus y el *deploy* de la aplicación SGS, se intentó un primer acceso a la aplicación. Al momento de ingresar el usuario y contraseña para acceder a el sistemas, se invoca a un *stored procedure* denominado "validuserserv". Dado que la primera intención es hacer funcionar la aplicación, este *stored procedure* no fue traducido de manera que se correspondiera con el código del original en Oracle, sino de manera tal de que devolviera los valores necesarios para permitir validar al usuario en cualquier situación. Aun así, dado que en el código del procedimiento se hace referencia a las tablas "wf_unidad", "stfar", "seguridad_ssae" y "dba_role_privs_ssae" y siendo que las tres primeras tablas estaban ausentes en el esquema generado por Genexus, se las creó siguiendo las definiciones existentes en la base de datos Oracle(ver fig. 4.1).

```
CREATE TABLE wf_unidad (
    clave numeric(6,0) NOT NULL, codigo character(3) NOT NULL,
    nombre character(100) NOT NULL
);

CREATE TABLE stfar (
    idfarmacia numeric(10,0) NOT NULL, codigo character(9) NOT NULL,
    n_pres numeric(10,0) NOT NULL, sicof numeric(10,0) NOT NULL,
    generico character(40) NOT NULL, descrip character(40) NOT NULL,
    forma_farm numeric(10,0) NOT NULL, unidad character(200) NOT NULL,
    empaque numeric(10,0) NOT NULL, cantemp numeric(10,0) NOT NULL,
    concentrac character(15) NOT NULL, labor numeric(10,0) NOT NULL,
    codbarra character(13) NOT NULL, onco_nro numeric(10,0) NOT NULL,
    id_odg numeric(10,0) NOT NULL, aux_odg numeric(10,0) NOT NULL
);

CREATE TABLE seguridad_ssae (
    grantee character varying(30) NOT NULL,
    granted_role character varying(30) NOT NULL
);
```

Figura 4.1: Creación de tablas faltantes al generar el esquema con GENEXUS

La tabla "dba_role_privs_ssae", ausente en los esquemas migrados y en el esquema "SSAEPROD" de Oracle, se creó a semejanza de la vista "dba_role_privs" existente en el esquema de "SYS" de Oracle(ver 4.2).

```
CREATE TABLE dba_role_privs_ssae (
    grantee character varying(30),
    granted_role character varying(30),
    admin_option character varying(3),
    default_role character varying(3)
);
```

Figura 4.2: Creación de tablas faltantes en el esquema principal

Como se observa, la principal utilidad de estas tablas es manejar los roles y controlar el acceso a la aplicación. El hecho que estas tablas no sean generadas por Genexus y que

se invoque a *stored procedures* de la base de datos desde el código generado por Genexus, brinda una idea de que tan importante son los *stored procedures* para esta aplicación.

4.3. Migración de Stored Procedures

Los *Stored Procedures* son una herramienta muy útil que brindan los RDBMS de hoy en día, con ellos se puede ejecutar código precompilado de forma rápida, permitiendo así acelerar los tiempos de ejecución de consultas o transacciones.

4.3.1. Stored Procedures MySQL

La migración de los *Stored Procedures* de Oracle a MySQL[8] no es del todo trivial, esto se debe principalmente a diferencias en la sintaxis de las sentencias, funcionalidades y a particularidades definidas de cada base de datos. Mientras que MySQL se adhiere al estándar ANSI/ISO SQL:2003, Oracle utiliza el lenguaje de programación PL/SQL.

Las principales diferencias al declarar un *stored procedure* en MySQL y Oracle[17] en el contexto del problema se dieron en:

- Definición de tipos de variables
- Diferencias entre Sentencias
- Declaración y uso Cursores
- Manejo de Excepciones

A continuación se detalla cada una de las diferencias.

Tipos de Variables

MySQL permite definir tres tipos de variables: variables locales, variables de sesión y variables de usuario. Las Variables Locales pueden ser definidas por convención al inicio de un *stored Procedure* o dentro de un bloque BEGIN...END y solo tienen validez dentro de este bloque o dentro de los bloques anidados. En caso de declarar dos variables con el mismo nombre en un bloque y en otro anidado, dentro del bloque anidado tendrá precedencia la variable definida allí. Antes de utilizar una variable se debe definir con la sentencia "declare", especificando el tipo de dato SQL. Las Variables de Sesión se declaran de la misma manera que las variables locales con la diferencia que se antepone el símbolo "@", de esta manera, tendrán validez dentro de la sesión del usuario y se borrarán de forma automática cuando se termine la sesión.

Por su parte PL/SQL permite definir variables dentro de un *stored Procedure* solo en la sección declarativa. A diferencia de lo que ocurre en MySQL, no existen variables de usuario o de sesión. En definitiva, todas las variables son locales. La precedencia al declarar dos variables de igual nombre dentro de bloques anidados opera de la misma manera que para MySQL.

Las variables en Oracle pueden ser de cualquier tipo definido en PL/SQL y una gran ventaja que existe sobre MySQL es que se puede hacer referencia a tipos definidos previamente en tablas o columnas de la base de datos. Utilizando los calificadores "%TYPE" y "%ROWTYPE" se puede indicar en forma implícita que el tipo de la variable a declarar se corresponde a otro tipo o tipo de columna definido anteriormente.

Al momento de realizar la migración en forma manual, es decir traducir el código de los *stored procedures* de PL/SQL a MySQL, el mayor problema encontrado se debió a los tipos de datos existentes. A modo de ejemplo, el tipo "VARCHAR2" de Oracle no existe en MySQL es así que se debió utilizar su similar "VARCHAR". Otro de los problemas fue el uso de los calificadores "%TYPE" y "%ROWTYPE" al momento de declarar las variables, como se dijo previamente no existe este tipo de calificadores en MySQL por tal motivo fue necesario observar la definición de los tipos de columnas de las tablas a las cuales hacía referencia y declarar explícitamente el tipo en cada uno de los casos.

Diferencias entre Sentencias

Cada lenguaje utilizado tiene sus bondades y particularidades, esto hace que existan sentencias y funcionalidades que no sean soportadas por el otro lenguaje. Un caso de ejemplo es la función "trunc", la cual no existe en mysql por lo que se tuvo que utilizar la función "datedif" (ver fig. 4.3).

```
-- en ORACLE
    v_dias:= trunc(sysdate) - trunc(v_FCH_DESPACHO);

-- en MySQL
    SET v_dias= DATEDIFF(CURDATE(),v_FCH_DESPACHO);
```

Figura 4.3: Diferencias ente asignaciones

Otra diferencia significativa encontrada es la función de concatenación de cadenas de texto. Mientras en Oracle y PostgreSQL se utiliza el operador "||", en MySQL debe invocarse a la función "CONCAT" de la siguiente manera "SET concatstr = CONCAT(str1, str2)".

Cursores

Los cursores permiten gestionar las sentencias select, es decir, un cursor devuelve un conjunto de registros de una sentencia sql. Tipicamente existen dos tipo de cursores, los cuales se describen a continuación:

- **Cursores implícitos**, utilizados para operaciones "SELECT INTO", son útiles básicamente cuando la consulta devuelve un único registro.
- **Cursores explícitos** son declarados y controlados por el programador. Se utilizan cuando la consulta devuelve un conjunto de registros. Ocasionalmente también se utilizan en consultas que devuelven un único registro por razones de eficiencia, son más rápidos.

Los cursores explícitos en PL/SQL tienen la particularidad de ser actualizables, es decir que cuando se está realizando un recorrido sobre el cursor se pueden modificar datos en la propia sentencia, en cambio los cursores explícitos de MySQL son solo de lectura, por lo que para actualizar los registros del cursor se debe tener una referencia al "rowid" del registro y con una sentencia "update" actualizar el valor.

Manejo de Excepciones

Las excepciones permiten manejar las alternativas que se pueden generar cuando se esta ejecutando el código, como puede ser cuando se espera que una consulta tenga resultados y no los tiene. Cada lenguaje implementa las excepciones a su manera. En el caso de MySQL, denomina a las excepciones como "Handlers" y en Oracle "Exceptions".

Si bien la funcionalidad es la misma, la figura 4.4 muestra las diferencias en la declaración de excepciones entre MySQL y Oracle.

```
-- en ORACLE

          DECLARE
            -- Declaraciones
          BEGIN
            -- Ejecución
          EXCEPTION
          WHEN NO_DATA_FOUND THEN
            -- Se ejecuta cuando ocurre una excepción de tipo NO_DATA_FOUND
           WHEN OTHERS THEN
            -- Se ejecuta cuando ocurre una excepción de un tipo no tratado
            -- en los bloques anteriores
          END;

-- en MySQL

        BEGIN
           DECLARE EXIT HANDLER FOR NOT FOUND
              BEGIN
                -- Se ejecuta cuando ocurre una excepción de tipo NO_DATA_FOUND
              END;
           DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
              BEGIN
                -- Se ejecuta cuando ocurre un error no específico
              END;
        -- Ejecución
  END;
```

Figura 4.4: Diferencias al declarar manejadores de errores

Cuando ocurre un error en Oracle, se ejecuta el bloque EXCEPTION, transfiriéndose el control a las sentencias del bloque. Una vez finalizada la ejecución del bloque de EXCEPTION no se continúa ejecutando el bloque anterior. A diferencia, MySQL permite definir el tipo de comportamiento con la condición EXIT — CONTINUE. La primera una vez que se ejecuta el bloque de la excepción no se continúa con la ejecución que la generó, mientras que la segunda continúa ejecutando el código.

4.3.2. Stored Procedures Postgres

Como primera versión, las funciones en Postgres se dejaron definidas, pero sin código activo, de manera de permitir el testing y traducir realmente aquellas necesarias.

Una de las dificultades encontradas para la migración de los *stored procedures* es la utilización de *savepoints* a los que se vuelve en caso de ser necesario hacer *rollback*. Dado que las funciones se ejecutan dentro de una transacción externa, hacer *rollback* o *commit* dentro, significaría abortar la función. Por otra parte, en el caso de los *rollbacks* llamados desde excepciones, estos se vuelven innecesarios porque Postgres los implementa automáticamente. En otros casos, podría ser necesario rediseñar los procedimientos[28]

Dado que la instrucción PRAGMA no existe en Postgres, aquellas funciones que utilicen la instrucción PRAGMA AUTONOMOUS_TRANSACTION deberían rediseñarse para no hacer uso de ella (Íbid).

En relación a la sintaxis, no se encontraron diferencias significativas entre Oracle y Postgres para los *stored procedures*.

4.3.3. El Proceso de Migración

Si bien la traducción de los *Stored Procedures* de Oracle a MySQL se puede hacer en forma manual, siguiendo las reglas previamente establecidas, se optó por realizar la migración probando una herramienta paga denominada SQLWays[9]. Esta herramienta permite convertir *stored procedures*, funciones, *triggers*, esquemas, datos, consultas SQL y *scripts* de un sistema de base de datos a otro. Entre otras funcionalidades permite convertir consultas SQL en Java, .NET, C/C++, PowerBuilder, Visual Basic y ASP. Tiene sentido utilizar esta herramienta cuando se debe convertir una considerable cantidad de rutinas o consultas en *scripts* o aplicaciones.

```
PROCEDURE "VALIDUSERSERV"
(pName_user in out SSAEMAEUSER.Name_user%type, pSerDsc in out servicio.serdsc%type, pUserValido in out varchar2)
is
-- <LocalVariable> <Datatype>;
vpstrServicio varchar2(200);
cantgrantee   NUMBER(5);
--itero       NUMBER(1);

CURSOR cgrantee (pppstrServicio varchar2) is
SELECT GRANTEE FROM DBA_Role_Privs_SSAE
WHERE GRANTEE LIKE 'SSAEPERF%' AND GRANTED_ROLE = pppstrServicio;
--rhora hora%rowtype; No es necesario para el for
--rhorfun chorfun%rowtype;
begin
--    <Statement>;
   puservalido   := 'N';
   vpstrServicio := 'SSAESERV' || pSerDsc;
   for rgrantee in cgrantee(vpstrServicio) loop
   begin
      select 1 into cantgrantee
      --SELECT count(*) into cantgrantee
      FROM DBA_Role_Privs_SSAE
      WHERE GRANTEE = pName_user AND GRANTED_ROLE = rgrantee.grantee;
      puservalido := 'S';
      exit;
   /* if cantgrantee > 0 then
         puservalido := 'S';
      end if;   */
      exception
         when too_many_rows then
            puservalido := 'S';
            exit;
         when no_data_found then
            null;
      end;
   end loop;
end validuserserv;
```

Figura 4.5: Stored Procedure definido en la base Oracle

SQLWays fue utilizado a modo de prueba, es decir con una versión *Trial* para convertir los *stored procedures* de Oracle a MySQL. Si bien en la mayoría de los casos, los resultados

obtenidos fueron correctos, se encontró que, en procedimientos complejos, el resultado de la migración realizada por SQLWays incorporaba variables innecesarias causando en algún caso salidas diferentes de lo esperado. En definitiva, este comportamiento, similar al de cualquier traductor mecánico, obliga a la revisión de los resultados de la traducción de los *stored procedures* migrados.

```
CREATE PROCEDURE VALIDUSERSERV
(INOUT pName_user VARCHAR(255), INOUT pSerDsc VARCHAR(255), INOUT pUserValido VARCHAR(4000))
-- SQLWAYS_EVAL# <Datatype>;
begin
-- SQLWAYS_EVAL# >;
   DECLARE vpstrServicio VARCHAR(200);
   DECLARE cantgrantee MEDIUMINT;
-- SQLWAYS_EVAL# UMBER(1);
   DECLARE pppstrServicio VARCHAR(4000);
   DECLARE NO_DATA INT DEFAULT 0;
   DECLARE SWV_CGRANTEE_GRANTEE VARCHAR(255);

   DECLARE CGRANTEE  CURSOR FOR SELECT GRANTEE FROM DBA_Role_Privs_SSAE
   WHERE GRANTEE LIKE 'SSAEPERF%' AND GRANTED_ROLE = pppstrServicio;
   DECLARE CONTINUE HANDLER FOR NOT FOUND SET NO_DATA = -1;
-- SQLWAYS_EVAL# No es necesario para el for
-- SQLWAYS_EVAL# un%rowtype;

   SET pUserValido   = 'N';
   SET vpstrServicio = CONCAT('SSAESERV',pSerDsc);
   SET pppstrServicio = vpstrServicio;
   OPEN CGRANTEE;
      SET NO_DATA = 0;
      FETCH CGRANTEE INTO SWV_CGRANTEE_GRANTEE;
      SWL_Label20:
      WHILE NO_DATA = 0 DO
      begin
         DECLARE EXIT HANDLER FOR 1328
         begin
            SET pUserValido = 'S';
         end;
         DECLARE EXIT HANDLER FOR NOT FOUND
         begin
            SET @SWV_Null_Var = 0;
         end;
         select 1 into cantgrantee
         -- SQLWAYS_EVAL# into cantgrantee
         FROM DBA_Role_Privs_SSAE
         WHERE GRANTEE = pName_user AND GRANTED_ROLE = SWV_CGRANTEE_GRANTEE;
         SET pUserValido = 'S';
         LEAVE SWL_Label20;
         /*SQLWAYS_EVAL# > 0 then
            puservalido := 'S';
         end if;   */
         end;
         SET NO_DATA = 0;
         FETCH CGRANTEE INTO SWV_CGRANTEE_GRANTEE;
      END WHILE;
      SET NO_DATA = 0;
   CLOSE CGRANTEE;
end;
```

Figura 4.6: Stored Procedure migrado con SQLWAYS

En la figura 4.5 se puede observar el procedimiento VALIDUSERSERV como se encuentra en el esquema de la base de datos Oracle mientras que la figura 4.6 muestra el resultado producido por SQLWays para MySQL tras migrar el *stored procedure* anterior.

Algunas de las particularidades observadas al traducir con la herramienta SQLWays se enumeran a continuación:

- **Líneas Comentadas**, SQLWays detecta en el código las líneas que se encuentran comentadas y las vuelve a comentar comenzando con "- SQLWAYS_EVAL#".
- **Utilización de nuevas variables**, debido a la incompatibilidad entre los lenguajes, en ciertas circunstancias es necesario agregar variables. SQLWays hace uso de algunas variables nuevas definiéndolas con el prefijo "SWV" utilizando el tipo de variables de sesión (anteponiendo el caracter "@").
- **Utilización de sentencias LEAVE**, el código generado contiene etiquetas que son invocadas con la sentencia "LEAVE", estas etiquetas son declaradas con el prefijo "SWL_Label", seguidas por un contador global a toda la migración.

4.4. Migración de Triggers

Los *Triggers* nos permiten ejecutar acciones en forma automática cuando ocurre un evento sobre una tabla, como pueden ser INSERT, UPDATE o DELETE. En el SGS los *triggers* tienen fundamentalmente la función de auditar las operaciones sobre la base de datos.

```
CREATE TRIGGER "FUNCIONARIO_JNTRG"
AFTER INSERT OR DELETE OR UPDATE ON FUNCIONARIO FOR EACH ROW
DECLARE
v_operation VARCHAR2(10) := NULL
BEGIN
IF INSERTING THEN
v_operation := 'INS';
ELSIF UPDATING THEN
v_operation := 'UPD';
ELSE
v_operation := 'DEL';
END IF;

IF INSERTING OR UPDATING THEN
  INSERT INTO FUNCIONARIO_JN (FUNID, FUNNOM, FUNAPE, FUNDOM, FUNTEL, FUNCI, FUNNROCAJAPROF, FUNNOCOBMSP,
FUNNOCOBCOM, aud_action, aud_date, aud_user)
VALUES ( :new.FUNID, :new.FUNNOM, :new.FUNAPE, :new.FUNDOM, :new.FUNTEL, :new.FUNCI, :new.FUNNROCAJAPROF,
:new.FUNNOCOBMSP, :new.FUNNOCOBCOM, v_operation, SYSDATE, USER);
  ELSE
  INSERT INTO FUNCIONARIO_JN ( FUNID, FUNNOM, FUNAPE, FUNDOM, FUNTEL, FUNCI, FUNNROCAJAPROF, FUNNOCOBMSP,
FUNNOCOBCOM, aud_action,aud_date,aud_user)
VALUES ( :old.FUNID, :old.FUNNOM, :old.FUNAPE, :old.FUNDOM, :old.FUNTEL, :old.FUNCI, :old.FUNNROCAJAPROF,
:old.FUNNOCOBMSP, :old.FUNNOCOBCOM, v_operation,SYSDATE,USER);
END IF;  --INSERTING OR UPDATING

END;
```

Figura 4.7: Ejemplo de trigger en PL/SQL

Los *triggers* de Oracle utilizados para la aplicación SGS, definen los tres eventos *INSERT*, *UPDATE* y *DELETE* en forma simultánea. Luego en la lógica, se establece qué acción se ejecutará de acuerdo al evento. La funcionalidad de Oracle, denominada predicados condicionales "INSERTING", "UPDATING" y "DELETING" permite consultar

y determinar el evento que disparó el *trigger*[21]. PostgreSQL opera de manera similar, pero en lugar de utilizar predicados condicionales, registra el evento que dispara el *trigger* en la variable "TG_OP", que puede asumir los valores "INSERT", "UPDATE" o "DELETE". Pero en MySQL no esta permitido definir más de un evento en el mismo *trigger*. Esto obliga a definir un *trigger* por cada evento dando por resultado que en MySQL se deban definir más *triggers* que los originales en Oracle.

Debido a lo mencionado anteriormente, en MySQL se renombran los *triggers* definidos para Oracle, los cuales mantienen el nombre inicial agregando al final una letra que identifica el evento a capturar.

```
CREATE TRIGGER FUNCIONARIO_JNTRG_I
AFTER INSERT ON FUNCIONARIO FOR EACH ROW
INSERT INTO FUNCIONARIO_JN (FUNID,FUNNOM,FUNAPE,FUNDOM,FUNTEL,FUNCI,FUNNROCAJAPROF,FUNNOCOBMSP,
FUNNOCOBCOM,aud_action,aud_date,aud_user)
VALUES ( new.FUNID, new.FUNNOM, new.FUNAPE, new.FUNDOM, new.FUNTEL, new.FUNCI, new.FUNNROCAJAPROF,
new.FUNNOCOBMSP, new.FUNNOCOBCOM, 'INS' ,NOW(), USER());

CREATE TRIGGER FUNCIONARIO_JNTRG_U
AFTER UPDATE ON FUNCIONARIO FOR EACH ROW
INSERT INTO FUNCIONARIO_JN (FUNID,FUNNOM,FUNAPE,FUNDOM,FUNTEL,FUNCI,FUNNROCAJAPROF,FUNNOCOBMSP,
FUNNOCOBCOM,aud_action,aud_date,aud_user)
VALUES ( new.FUNID, new.FUNNOM, new.FUNAPE, new.FUNDOM, new.FUNTEL, new.FUNCI, new.FUNNROCAJAPROF,
new.FUNNOCOBMSP, new.FUNNOCOBCOM, 'UPD' ,NOW(), USER());

CREATE TRIGGER FUNCIONARIO_JNTRG_D
AFTER DELETE ON FUNCIONARIO FOR EACH ROW
INSERT INTO FUNCIONARIO_JN (FUNID,FUNNOM,FUNAPE,FUNDOM,FUNTEL,FUNCI,FUNNROCAJAPROF,FUNNOCOBMSP,
FUNNOCOBCOM,aud_action,aud_date,aud_user)
VALUES ( old.FUNID, old.FUNNOM, old.FUNAPE, old.FUNDOM, old.FUNTEL, old.FUNCI, old.FUNNROCAJAPROF,
old.FUNNOCOBMSP, old.FUNNOCOBCOM, 'DEL',NOW(), USER());
```

Figura 4.8: Traducción para MySQL del trigger de la figura 4.7

En la fig. 4.7 se puede observar como un *trigger* definido en el esquema de Oracle es migrado al esquema Mysql, dando como resultado los *triggers* mostrados en la fig. 4.8.

Salvaguardando lo descripto anteriormente, no existen mayores inconvenientes para migrar el código de los *triggers* definidos en el SGS de Oracle tanto a PostgreSQL como a MySQL.

4.5. Migración de datos

La migración de datos fue planificada tomando en cuenta las siguientes restricciones:

- gran cantidad de tablas
- algunas pocas tablas con una gran cantidad de registros
- necesidad de desarrollar un proceso adaptable a todos los motores evaluados
- necesidad de que el proceso fuera reanudable luego de cualquier interrupción

A los efectos de contar con un proceso práctico, dada la gran cantidad de datos y el tiempo que su migración insume, surgió la idea de generar archivos temporales con los datos de la base origen (Oracle), y a partir de estos, generar los datos en los motores evaluados. Además, los archivos temporales se pueden generar de una forma

relativamente rápida, para luego experimentar en la mejor forma de insertarlos en el motor específico, esto es, eliminando índices y claves foráneas de las tablas, para recrearlos luego de insertados los datos, o simplemente insertar sin ninguna alteración en la definición de las tablas. La generación de archivo temporales tiene dos desventajas: mayor consumo de recursos de espacio disco y mayor tiempo de proceso que volcar directamente de una base a otra sin intermediarios. La primera desventaja no representa inconvenientes debido al bajo costo del almacenamiento en disco. La segunda representa un contratiempo mayor, pero la ventaja de que el proceso sea particionable (y por lo tanto repetible y mejorable) tiene mucho mayor peso en la decisión.

De acuerdo a esto, la mejor alternativa fue desarrollar un script capaz de conectarse a Oracle, volcar datos en archivos, y luego insertarlos en los diferentes motores.

Python fue elegido como el lenguaje para desarrollar los scripts de migración, debido a que: a) cuenta con una interfaz de conexión genérica a todos los motores considerados (Python DB-API interface), b) es un lenguaje de scripting con una sintaxis muy sencilla, y c) es soportado por la mayoría de los sistemas operativos habituales y especialmente por Linux.

4.5.1. Volcado de datos

.

Como se mencionó, existe una gran cantidad de tablas, y algunas de ellas con varias decenas de millones de registros. Durante el proceso de volcado de datos, se eligió generar un conjunto de archivos por cada tabla del esquema de origen (Oracle, como ya se mencionó). Cada archivo contiene una cantidad configurable de registros, y cada línea del archivo representa un registro. Una línea por registro facilita los algoritmos de volcado y de cargado de datos. Archivos relativamente pequeños facilitan la tarea de cargar datos, confirmando los datos insertados de cada archivo en la tabla destino. Si el proceso de carga se interrumpe (escenario muy probable debido a imprevistos), es necesario comenzar a partir del último archivo cargado. El diseño de la base de datos origen no presenta integridad referencial, por lo tanto no será necesario reflejar relaciones de integridad en los archivos de volcado. Es muy importante considerar este hecho ya que facilita en gran forma el procedimiento. Todas las tablas tienen clave primaria, esto permite que el volcado se realice en orden ascendente según la clave.
Estructura de archivos:
Dada una tabla, se genera:

- un archivo que contiene una lista de nombres de sus columnas a modo de descriptor de datos
- n archivos de datos conteniendo una línea por registro de la tabla

Cada línea del archivo es una serialización Python de los datos de un registro.
Archivo descriptor:

```
nombre:
<tabla>_descriptor
contenido:
representación de un diccionario de Python de esta forma:
{'table_fields': ['<campo_1>', '<campo_2>', .. , <campo_n>]}
```

Archivos de datos:

n archivos de datos según la cantidad de registros de la tabla y la cantidad configurada de registros por archivo.
Cada línea de un archivo contiene una representación de una tupla de la tabla:

```
('<valor_1>', '<valor_2>', .. , <valor_n>)
```

Esta representación resultó adecuada para programar los algoritmos de volcado e incluso inspeccionar datos visualmente cuando se hizo necesario encontrar errores.

4.5.2. Carga de datos

.

La carga de datos se realiza utilizando el script desarrollado para tal fin. (ver Anexo B "Scripts de Migración").

El procedimiento básico es leer el archivo de volcado de a una línea por vez, traducir la línea a un objeto Python y luego insertarlo en el nuevo motor utilizando la interfaz de Python con el motor considerado. Se comienza una transacción y se confirma luego de que se haya procesado una cantidad de registros definida. Si la carga de datos se interrumpiera, es posible recorrer el archivo de volcado hasta alcanzar el último registro insertado y confirmado.

4.5.3. Procedimiento de migración

Debido a la misión critica del sistema, no es posible que los datos del SGS estén fuera de línea durante mucho tiempo. Además, cuando el sistema está funcionando normalmente, recibe gran cantidad de información en un corto período de tiempo.

La cantidad de datos es tal que no es posible migrar el 100 % en una ventana de tiempo con el sistema fuera de línea. Levantar un respaldo físico de Oracle lleva aproximadamente 48 horas, según el DBA de ASSE, por lo tanto cualquier proceso de migración que involucre operaciones a nivel lógico, como las que se plantean en este procedimiento, consumiría un tiempo inaceptable (más de 8 horas) de fuera de línea para el SGS. La estructura de datos binarios de Oracle no está documentada y no existe forma razonable dentro del alcance de este proyecto de trabajar en la migración a bajo nivel.
Se definen dos procedimientos de migración:

1. **Migración para evaluación:** Es la migración de datos necesaria para la evaluación del sistema. No existen restricciones importantes de tiempo de procesamiento. Esta migración fue realizada en este proyecto.
El procedimiento para efectuar la migración básica de datos es el siguiente:

a. Ejecutar el *script* de volcado de datos (ver detalles en apéndice B "Scripts de migración"),

b. Ejecutar el *script* de carga de datos (ver detalles en apéndice B "Scripts de Migración").

2. **Migración para producción:** Es la migración de datos tal como debería ser ejecutada en un ambiente de producción, dando como resultado el conjunto de datos completo y consistente al que accedía el sistema hasta el momento del cambio.

La migración para producción requiere de un tratamiento más complejo. No es posible realizar la migración completa con el sistema fuera de línea debido al tiempo que insumiría.

Es necesario encontrar una estrategia para migrar datos minimizando el tiempo de fuera de línea (8 horas).

Se propone dividir el conjunto de datos en dos partes. Un subconjunto de tablas con relativamente pocos datos, y por otro lado el subconjunto de tablas 'grandes' (decenas de millones de registros). Del relevamiento surgió que las tablas grandes no son críticas para el sistema, por lo que el SGS puede funcionar correctamente en forma provisoria aunque estas tablas no contengan todos los registros anteriores.

Analizando esta realidad, se propone el siguiente procedimiento de migración para producción:

1. Migrar datos de tablas 'grandes'. El sistema original se mantiene en línea y por lo tanto se presenta un conflicto.
2. Desconectar al SGS del RDBMS original definitivamente. El SGS estará fuera de línea.
3. Volcar datos de tablas 'chicas' (realizado en un tiempo menor a 2 horas).
4. Cargar datos de tablas 'chicas' para el nuevo motor (realizado en un tiempo menor a 6 horas).
5. Montar el SGS sobre el nuevo RDBMS. El SGS volverá a funcionar en forma provisoria.
6. Cargar datos volcados de tablas 'grandes' (volcados en el punto 1). El SGS funciona en forma completa definitivamente.

El conflicto se presenta en el punto 1, ya que durante el volcado de las tablas de gran tamaño, el sistema está en funcionamiento y recibiendo cambios que quedarán fuera de la migración. Recordar que el volcado se realiza siguiendo un orden de la clave primaria.

Ejemplo de conflicto:

Supóngase que la tabla volcada tiene como clave primaria al documento de identidad y que el proceso de volcado ha superado los registros con valores de 2 millones. Si en ese momento un paciente con documento 1 millón es referenciado y genera una modificación sobre la tabla, entonces el registro volcado al archivo es inválido. Esta clase de cambios quedarán fuera de los archivos de volcado, por lo tanto debe agregarse un mecanismo al sistema original que haga posible migrar todos los datos.

Se propone el siguiente mecanismo:

Poco antes de comenzar el volcado y para cada tabla de las llamadas grandes, agregar un *trigger* y una tabla auxiliar en el esquema original. El *trigger* insertará en la tabla auxiliar los valores correspondientes a la clave primaria de la tabla original, y un *flag* para indicar si se trata de una inserción, modificación o eliminación. De esta manera se genera un historia de lo que sucede con los datos durante el volcado. Este registro será utilizado para decidir si un registro se carga o no en el nuevo motor. Como se vio, la carga se realiza a partir de los datos volcados en un archivo.

Evitar ingresar registros eliminados:

Antes de insertar el registro, se verifica que este no haya sido eliminado, consultando la tabla auxiliar correspondiente. De ser así se encontrará en esta una marca de eliminado.

Evitar ingresar registros desactualizados:

De igual forma, se verifica el registro por modificación. Si hubiera sido modificado, ignorar el registro del archivo y tomar como origen el de la tabla original. Este registro es accesible porque se mantiene la identificación por su clave primaria.

Evitar perder registros:

Los registros ingresados al sistema luego del volcado de datos estarán marcados en la tabla auxiliar, de forma tal que un recorrido de esta seleccionando los registros marcados con el *flag* de inserción dará como resultado un conjunto de registros que se deben insertar en el nuevo RDBMS. Simplemente para cada uno de ellos, se seleccionan los datos originales a través de sus claves primaria y se insertan en el nuevo sistema.

4.6. Resumen

El proceso de aprendizaje sobre la migración tomó mucho tiempo debido al porte del sistema. Tablas con muchos millones de registros, más de cien *stored procedures* y *triggers* hicieron que la tarea fuera complicada. Se decidió enfrentar la migración a bajo nivel, esto es, realizando los cambios manualmente o construyendo herramientas personalizadas. El motivo de esta decisión está ligado al objetivo del proyecto respecto de enriquecer el conocimiento existente sobre RDBMS FOSS.

En este sentido se obtuvo información sobre gran cantidad de detalles particulares a cada RDBMS evaluado. Durante el relevamiento de herramientas, no se encontró ninguna capaz de transformar la base de datos en forma íntegra, aunque sí algunas proveyeron de ayuda al momento de generar estructuras que después hubo que corregir manualmente. La migración fue satisfactoria dentro del marco de los objetivos tanto para MySQL como para PostgreSQL, ya que como se verá en el próximo capítulo, el SGS funcionó correctamente dentro de los casos de uso planteados para estos RDBMS.

Capítulo 5

Testing

5.1. Introducción

Este capítulo da cuenta de los casos de uso probados, de su selección y de las dificultades encontradas. Al comienzo del capítulo, se especifican los casos de uso que se tomarán en cuenta y se seleccionan juegos de datos para la realización de las pruebas. Luego de esto se evalúan los resultados obtenidos y se sacan conclusiones

5.2. Casos de uso testeados

De parte de ASSE, se cuenta con un documento que especifica una selección de catorce casos de uso elegidos por su incidencia en el rendimiento del Sistema tanto por la frecuencia de uso como por la exigencia al servidor. Estos casos de uso, se clasifican en alta, media y baja frecuencia de uso.

De este conjunto de casos de uso, se seleccionaron cuatro para las pruebas preliminares. De ellos, uno es de baja frecuencia, otro de alta frecuencia y los dos restantes de frecuencia media.

El primer caso elegido es la Programación de Oferta de Servicios. El documento de ASSE, indica que *"permite programar las agendas de los diferentes servicios definiendo días y horarios de atención, intervalos de consulta y funcionarios asignados a roles"*. Este caso está clasificado como de baja frecuencia, no especificándose ni cantidad de transacciones ni tiempo normal de uso.

El segundo caso elegido es la Atención de Solicitud de Consulta. La descripción de ASSE señala que *"permite identificar al usuario, actualizar sus datos patronímicos y de georeferencia, y crear una solicitud de consulta para el servicio, médico y fecha/hora que el usuario elija"*. Este caso de uso, presenta una cantidad de transacciones de 1225 entre 9 y 11 hs. con una cantidad de 59 usuarios. En un escenario normal, tiene una duración de 40 segundos.

El tercer caso es similar al anterior, pero de frecuencia media. Consiste en la Atención de Solicitudes de Emergencia. Presenta 111 transacciones entre 9 y 11 hs. con una cantidad de 10 usuarios y un tiempo normal de uso de 30 segundos.

El cuarto y último caso es el censo de internaciones. También clasificado como de frecuencia media, la documentación lo presenta como el caso de uso que *"permite consultar*

el estado de ocupación de un centro hospitalario dado, mostrando ordenado por sala y cama, el detalle de las internaciones activas con los datos del paciente y su diagnóstico, permitiendo hacer búsquedas por nombre y especialidad entre otros filtros". Aparece con 60 ejecuciones entre 9 y 11 hs. con una duración normal de 6 segundos.

5.3. Resultados obtenidos

Cada uno de estos cuatro casos de uso fue ejecutado contra la base en MySQL y en PostgreSQL, evaluándose los resultados. Se presentan a continuación el desarrollo de la ejecución de cada uno de los casos de uso antedichos. Como se ha mencionado, MonetDB no pudo se conectado a forma razonable, por lo tanto no forma parte de estos casos de prueba.

5.3.1. Programación de Oferta de Servicios

Este caso de uso consiste en la creación de la agenda de los médicos. Los datos de ingreso son las horas de comienzo y fin de la consulta, la duración asignada por paciente y el médico en cuestión.

La figura siguiente muestra el primer paso de este caso de uso:

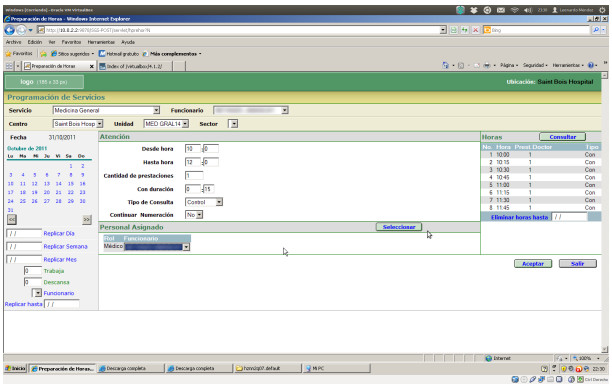

Figura 5.1: Captura de pantalla: Ingreso de datos en Programación de Oferta de Servicios

Al utilizar PostgreSQL como motor de base de datos, se detectó que tras el ingreso de los datos, se disparaba una excepción que pudo identificarse como un *bug* en el código Java generado por Genexus. Dado que este problema no se presentaba en la versión original para Oracle ni en la versión compilada para MySQL, se apuntó a la búsqueda de una solución en el código Java. Tras la decompilación de las clases involucradas se encontró un error en una llamada a la función "registerOutParameter" en la que se confundían los tipos de datos "Numeric" y "Bigint", Para resolverlo, se corrigió el error cambiando el registro del parámetro de salida y se volvieron a compilar las clases. Tras esto, pudo completarse el caso de uso. El detalle del error encontrado y la corrección del mismo se desarrollan en detalle en el apéndice 4.

En MySQL, se presenta un problema por la falta de la existencia de "*sequences*". Dada la necesidad de llevar registro de la numeración y de una implementación que utiliza las funciones usuales de "*sequences*", se evaluó que la propiedad autoincremental de MySQL no resulta suficiente para cubrir los requerimientos del sistema, por lo cual se emuló el

funcionamiento de las "*sequences*" creando las funciones necesarias para su uso. La tabla utilizada y las funciones creadas se desarrollan en el apéndice 5.

A continuación se ve la pantalla final del caso de uso, en la que se muestra el guardado exitoso de la agenda del médico.

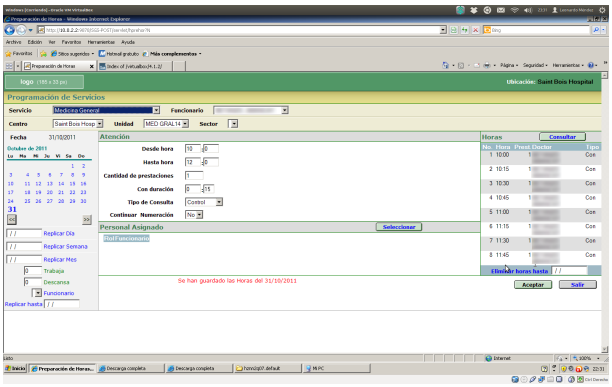

Figura 5.2: Captura de pantalla: Resultado de la Programación de Oferta de Servicios.

5.3.2. Atención de Solicitud de Consulta

Este caso de uso permite asignar las horas programadas anteriormente a los usuarios del sistema de salud. Una vez ingresados los datos del usuario o recuperados de la base de datos, el sistema permite asignarle alguna de las horas programadas.

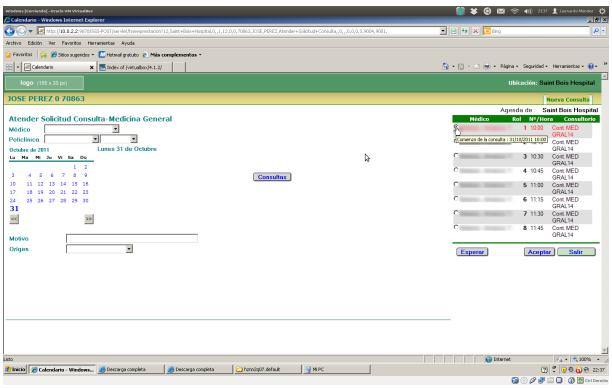

Figura 5.3: Captura de pantalla: Atención de Solicitud de Consulta

El ejemplo que se muestra en las figuras 5.3 y 5.4, se ingresó un nuevo usuario indocumentado, con el nombre José Pérez y se le otorgó el primer número programado en el ejemplo anterior. En la figura de arriba se puede ver la pantalla correspondiente al ingreso de datos. Finalmente, el sistema genera un ticket que se imprime y se entrega al usuario con toda la información de la consulta asignada.

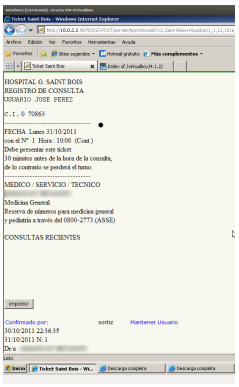

Figura 5.4: Captura de pantalla: Atención de Solicitud de Consulta

Este caso presentó el mismo problema en el código generado por Genexus, que se señalara en la sección anterior y que se desarrolla en detalle más adelante.

5.3.3. Atención de Solicitud de Emergencia

Este caso, si bien similar al anterior, presenta dos diferencias. En primer lugar, no se necesita vincular al paciente con la agenda de un médico, pues el paciente ya está en espera de la consulta. En segundo lugar se pide al usuario del sistema que, como dato adicional, ingrese el motivo de consulta.

En el ejemplo que se muestra a continuación (figura 5.5), se seleccionó un usuario por el número de documento de identidad y se lo ingresó a la Emergencia con un diagnóstico presuntivo de Apendicitis.

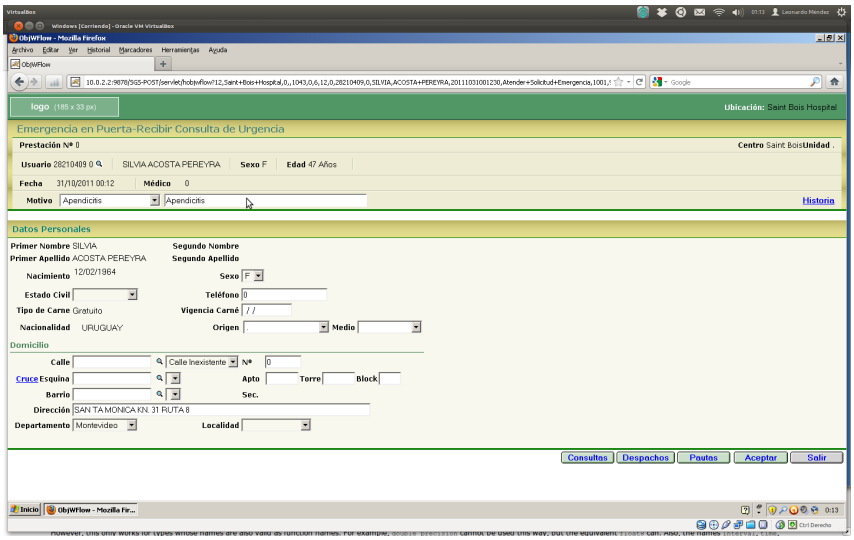

Figura 5.5: Captura de pantalla: Atención de Solicitud de Emergencia

Al aceptar, el sistema genera un ticket y el usuario queda incluido en la lista de pacientes a ser atendidos en emergencia.

El caso de uso de atención de solicitudes de emergencia no presentó ningún problema funcional al ser evaluado y respondió en los tiempos esperados.

5.3.4. Censo de Internaciones

El censo de internaciones, reporta el estado de uso de las camas del centro permitiendo realizar filtrados por unidad dentro del centro, por documento del usuario, por el servicio médico utilizado y según las camas estén libres u ocupadas.

La figura 5.6 muestra el resultado al acceder al Censo de Internaciones para el Hospital Saint Bois.

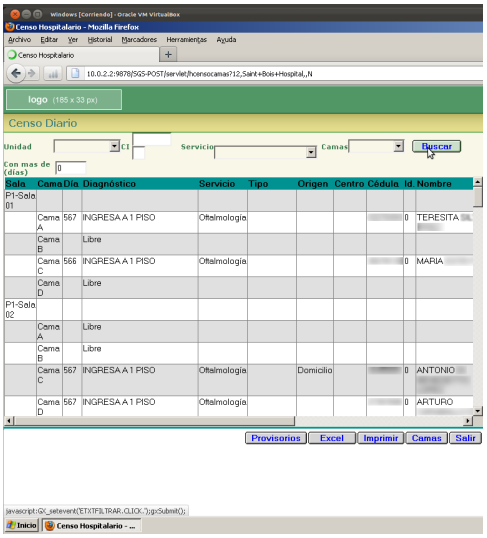

Figura 5.6: Captura de pantalla: Censo de Internaciones

A continuación se muestra el resultado tras filtrar por unidad (figura 5.7).

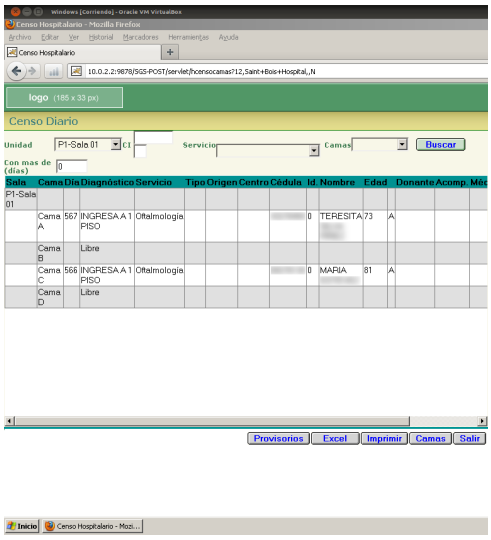

Figura 5.7: Captura de pantalla: Censo de Internaciones

Este caso tampoco presentó problemas en las pruebas. Responde de manera adecuada en cualquiera de los dos motores utilizados.

5.4. Dificultades

Se destacan dos, entre las dificultades encontradas; la primera vinculada con el testing y la segunda con la puesta en funcionamiento de la aplicación.

Acerca de la primera, la dificultad consistió en la imposibilidad de realizar pruebas de stress sobre el sistema una vez migradas las bases de datos. Si bien ASSE cuenta con algunas pruebas automatizadas, no hubo posibilidades de adaptarlas para hacerlas correr sobre los nuevos motores. Las pruebas realizadas, por tanto, se limitaron a garantizar que el sistema fuera utilizable para un subconjunto de casos de uso seleccionados.

Con respecto a la puesta en funcionamiento de la aplicación, los problemas principales se observaron en el uso *as is* de la aplicación compilada por Genexus. En primer lugar, se observó que no todos los *stored procedures* respondían correctamente cuando eran llamados. Este problema, detectado en un inicio con los *stored procedures* vinculados a la validación de usuarios encontró solución modificando el orden de los parámetros en las llamadas de Genexus. Colocando los parámetros de salida al comienzo, los *stored procedures* funcionaron correctamente.

Un segundo problema está vinculado con el código Java generado por Genexus. Se encontró que el código generado disparaba una excepción que imposibilitaba el correcto funcionamiento de algunos de los casos de uso seleccionados para las pruebas. Como se señaló en 5.3.1 y se desarrolla en detalle en D, la solución obligó a la decompilación, corrección y recompilación de la clase Java que presentaba el error.

5.5. Resumen

La dificultad de ejecutar los Casos de Uso planteados radicó principalmente en la poca información que el sistema proveía ante errores. De cualquier forma, los errores fueron

detectados y solucionados dando como resultado una interacción entre SGS y los DBMS (PostgreSQL y MySQL) adecuada respecto al comportamiento esperado.

Capítulo 6

Conclusiones

En los objetivos, se plantea la pregunta si es factible migrar la base de datos Oracle hacia una base FOSS para servir de *back end* del sistema "SGS". El trabajo no solo pretende responder en forma afirmativa o negativa a la pregunta sino que busca, además recolectar información relevante para ASSE y que no se circunscriba estrictamente al contexto de la aplicación "SGS".

Podemos afirmar como primera conclusión que la migración resultó exitosa tanto para el caso de MySQL como para el caso de PostgreSQL. Tras el proceso de migración, visto en el capítulo 4, las pruebas y los ajustes realizados, la aplicación terminó siendo funcional para los dos motores mencionados anteriormente. Se realizaron pruebas de uso manuales sobre la aplicación, para las que ambos motores respondieron satisfactoriamente (capítulo 5). En esta área, quedaría pendiente la realización de tests automatizados que permitieran evaluar de manera más rigurosa el rendimiento de los nuevos motores.

Esto quiere decir que la respuesta a la pregunta "¿es posible migrar y luego mantener el SGS?" debería, desde el punto de vista estrictamente técnico, responderse afirmativamente, tanto para MySQL como para PostgreSQL.

De la investigación también se obtuvo información relevante en los siguientes grupos de funcionalidades:

- Respaldos y recuperación ante fallas
  - respaldos en caliente,
  - disponibilidad de la base de datos ante fallas de software y/o hardware,
  - complejidad en el procedimiento de recuperación de datos frente a diversos escenarios, incluyendo tiempos de estas tareas para el volumen manejado por el sistema.

  Dado que tanto el XtraBackup de Percona, como el respaldo de PosgreSQL y su uso mediante el script pg_hotbackup admiten la recuperación de la base de datos a un punto determinado en el tiempo, ambos motores responden adecuadamente ante una falla crítica, mientras se mantengan seguros tanto los respaldos como los logs lógicos. En ninguno de los dos casos, el procedimiento reviste una alta complejidad (sección 3.3).

- Estadísticas

  - actualización de estadísticas
  - modificación del comportamiento del optimizador mediante la manipulación de estadísticas

  MySQL no presenta posibilidades de modificación de las estadísticas del optimizador. En PostgreSQL, existe la posibilidad de modificar las estadísticas, accediendo directamente a las tablas en las que se guarda la información. (3.6)

- Ajustes de Configuración
  - modificación del comportamiento de la base de datos mediante variables del sistema
  - elección de los niveles de aislamiento referente al manejo de transacciones

  MySQL y Postgres permiten realizar ajuste de configuración a los motores de acuerdo a las necesidades de la aplicación a la cual se encuentre sirviendo.

- Particionamiento

  Si bien actualmente la base de datos no contiene tablas particionadas, se pude observar que existen tablas que contienen decenas de millones de registros. El rendimiento en el acceso a los datos de dichas tablas podría mejorarse mediante el particionado.

Dada la situación generada luego de la migración, se abren perspectivas en algunas de las líneas de trabajo requeridas en los objetivos del proyecto. En primer lugar, podemos destacar una potencial ventaja comparativa importante en las líneas de la alta disponibilidad, la escalabilidad y la distribución de la carga. Al prescindir del costo de la licencia por servidor de base de datos instalado, se abre la posibilidad de generar clústeres de servidores sin añadir más costo extra que el de su configuración e implementación. Tanto PostgreSQL con el "middleware" pgpool[36] como MySQL con "MySQL cluster"[1] presentan alternativas viables para la alta disponibilidad. Por otra parte, se posibilita el particionado de bases de datos en diferentes servidores, que mediando una planificación adecuada puede permitir una distribución de la carga mejor gestionada que en un único servidor centralizado.

Una conclusión menos directa del trabajo realizado, abre la posibilidad de la implementación de servidores "open/free" en paralelo con los servidores existentes sin necesidad de realizar una migración total. Podría así, replicarse periódicamente la información residente en el servidor principal a un servidor secundario encargado de atender las consultas gerenciales y los procesos más demandantes de recursos.

Capítulo 7

Aportes y trabajos a futuro

7.1. Aportes

Surge como primer aporte de la migración realizada, la creación de los scripts que permiten la realización de la migración. Dado que la migración de los datos se desarrolla en el contexto de Python como lenguaje elegido, los mismos scripts pueden reutilizarse para migrar entre distintos motores, cambiando solamente los conectores por los de las bases de datos que se pretenda utilizar. El funcionamiento de estos scripts se asemeja al procedimiento utilizado por SQLWays para la migración de datos. En primer lugar se baja la información de la base de datos de origen a archivos de texto plano y, en segundo lugar se levanta la información de estos archivos al motor de destino. Para realizar la migración completa sobre una base de datos en funcionamiento, una vez realizada la migración, deberá detenerse el motor de origen para calcular el diferencial de datos por migrar, realizar el movimiento de dichos datos y levantar el motor de destino con la información completa. En el Anexo B se presenta el código de la versión final de los scripts que se utilizó para la migración.

En segundo lugar, otro aporte es el bug encontrado en el código java generado por Genexus para el funcionamiento con Postgres. En el Anexo D se desarrolla la estrategia utilizada para encontrar el error en el código, traduciéndose los resultados en un informe realizado para ASSE, de manera de que se pudiera reportar el bug a Artech.

7.2. Difusión

El presente proyecto de grado fue seleccionado y presentado en el marco de las Jornadas de Informáticos de la Administración Pública (JIAP) del año 2011, debido a su aporte en la evaluación de tecnologías FOSS aplicadas en la administración publica como lo es ASSE. [I]

7.3. Trabajos a Futuro

Como trabajos pendientes se destacan dos. En primer lugar, se podría buscar optimizar el tiempo en que la base de datos debe permanecer "off-line" durante la migración. Como

[I] http://www.asiap.org/AsIAP/index.php/jiap/jiap-anteriores/565

vimos anteriormente, en el proceso de migración, la mayor parte de los datos pueden migrarse con la base *on-line*. Sin embargo, para calcular los diferenciales de datos que puedan haberse modificado durante la migración y sincronizar definitivamente el motor de destino con el motor de origen, se hace necesario detener el motor de origen. Una posible manera de realizar este procedimiento consiste en hacer uso de alguna herramienta ad-hoc que permita el análisis de los logs del motor de la base de datos de origen. En el caso concreto de Oracle, surge como posibilidad a estudiar la herramienta GoldenGate [22] Mediante el uso de los logs, uno podría plantearse la situación ideal de que una vez migrada la base de datos, el diferencial sea procesado "on-line" por la herramienta en cuestión de manera de que las bases queden sincronizadas. De esta manera, el tiempo que se tardaría en pasar de un motor a otro podría, en teoría, ser prácticamente nulo. Si bien esta es una situación ideal, no debería descartarse a priori.

Por otro lado, la documentación existente acerca de los motores columnares hace que aparezcan como bastante prometedores, sobre todo a la hora de evaluar el desempeño. Si bien con la versión utilizada de MonetDB fue poco lo que se pudo avanzar, la existencia de nuevas versiones y la posibilidad de evaluar otros motores columnares queda como trabajo pendiente.

Anexos

Anexo A

Localización de puestos

A.1. Contexto

La aplicación necesita conocer la ubicación del puesto para habilitar el contexto de producción (consultas, médicos, stocks) del centro o policlínica que está operando. La ubicación del puesto está dada por dos datos: Centro (obligatorio) y Unidad (opcional) Centro: refiere al centro de asistencia (Hospital Pasteur, Centro de Salud del Cerro, Hospital de Ojos) Unidad: refiere a la policlínica dependiente del centro (IMM-Aquiles Lanza, IMM-Barrio Sur, Cerro-Entre Vecinos)

A.2. Situación Actual

Existe un programa que genera cookies para cada puesto que necesita identificarse. Para ubicar el puesto, previamente al acceso al SGS, se ejecuta un programa (cookie.exe) que genera una cookie para el servidor de aplicación con el nombre del puesto entre otros datos. Esta cookie se crea con cierta vigencia para no volver a requerir su ejecución. En el proceso de login, luego de autenticado el usuario, la aplicación busca la cookie con el nombre del puesto y accede a la BD para verificar que el puesto existe y determinar su ubicación (centro[unidad]). Si no encuentra la cookie o no encuentra el puesto en la BD, devuelve un mensaje de "Puesto no localizado.[ei]nterrumpe el login. Si encuentra la cookie y el puesto, depliega la ubicación del puesto en el banner del sistema (a la derecha) y posiciona todos los casos de uso de producción

A.3. Problemas que se presentan

Las cookies se generan solamente para Internet Explorer, por lo que intentar usar otro navegador supone la importación o generación de nuevas cookies. Un usuario podría utilizar las cookies generadas para transportarlas a otro puesto y utilizar un contexto de producción que no corresponde al lugar en el que se encuentra.

A.4. Requerimientos relevados

Se necesita identificar el puesto. Existen puestos que tienen dirección IP fija que podrían utilizarla para identificar su localización. Otros puestos trabajan con dirección

IP variable y también necesitan ser localizados. Se requiere que el sistema de localización de puestos funcione en plataformas “Open/Free”. Si bien no se presentan requerimientos explícitos con respecto a la seguridad, se ha señalado que no es un tema que debiera dejarse de lado sin más. Es deseable reducir al mínimo las posibilidades del usuario de seleccionar la ubicación del puesto de trabajo.

A.5. Esquema de soluciones propuestas

Mantener el esquema de funcionamiento de aplicación actual agregándole la generación de "cookies" para el resto de los navegadores de uso habitual.	* No se necesita invertir prácticamente ningún esfuerzo en desarrollo. * Soluciona el acceso desde plataformas "open/free" * No es necesario hacer modificaciones en el servidor.	* Mantiene los potenciales riesgos de seguridad actuales. * No es escalable: nuevos navegadores quedarían fuera de la solución.
Utilizar un proxy que redirija las consultas de los usuarios según su ubicación. El proxy se puede instalar en cada estación de trabajo o si se trata de una red local en un servidor local establecido.	* El acceso al sistema es transparente al usuario. * El usuario no puede cambiar su localización. * En el caso de una red local, se uniformiza el acceso al servidor central a través del proxy por parte de las estaciones de trabajo conectadas a la red local.	* Genera un nuevo punto posible de falla. * Debe instalarse un proxy en las máquinas de usuarios móviles. * Podría requerir modificaciones en el servidor si se quisieran permitir subdominios de la forma http://unidad[centro].rap.gub.uy
Utilizar un sistema de localización por dirección IP permitiendo a los usuarios móviles elegir la ubicación desde la que se conectan.	* Utiliza la dirección IP asociada a la red para identificar la localización. * No se necesitan configuraciones especiales para los usuarios que utilizan direcciones fijas ni para los usuarios móviles.	* Permite el error del usuario al elegir ubicación en caso de IP variable. * Requiere modificaciones en el sistema de acceso para permitir la selección del Centro[unidad] * Requiere modificaciones en el servidor.

A.6. Desarrollo de soluciones

A.6.1. Generación de cookies para diferentes navegadores

Costo estimado:
Investigar formato de cookies del navegador: 8 horas.
Escribir aplicación que genere las cookies: 12 horas.
Testing de la aplicación: 4 horas
Instalar programa en un cliente: 1 hora.

A.6.2. Proxy

Consiste en la instalación de un proxy al que el cliente se conecta para acceder al sistema. El proxy, sabiendo cuál es el sitio en el que está instalado, debería redirigir la consulta hacia el sistema, enviando la información de la localización, bien bajo forma de subdominio o enviando la información pertinente a los efectos de localizar al Centro[Unidad] conectado.

Costo estimado:
Investigación y elección del proxy: 8 horas.

Testing del proxy: 4 hora.

Instalación y configuración del proxy: 1 hora.

Modificación del servidor para recibir la ubicación: 4 horas.

Testing del servidor: 2 horas.

Costo de identificar y solucionar un error en caso de que el proxy dejara de funcionar por alguna razón: no cuantificable

A.6.3. Reconocimiento de IPs fijas, selección por parte del usuario para IPs variables

Debería permitir a los usuarios que manejan más de un puesto de trabajo, la posibilidad de elegir el puesto a utilizar para la conexión. En el caso en el que el usuario que se conecta tiene múltiples localizaciones asignadas, el sistema debería saber cuáles son y presentarle al usuario la posibilidad de elegir cuál va a utilizar en la conexión de turno.

Costo estimado:

Mapear las ubicaciones con las direcciones de IP en la base de datos: depende de la cantidad y la disponibilidad de la información.

Modificación del servidor para buscar la ubicación a partir de la dirección IP: 4 horas.

Testing: 2 horas.

Modificación en el sistema cliente para permitir elegir la ubicación en caso de IP variable: 4 horas.

Instalación de un cliente: 1 hora.

Costo de corregir datos si el usuario selecciona una ubicación errónea: no cuantificable.

A.6.4. Cliente notificador de IP y ubicación

En este esquema, el cliente envía cada cierto lapso la asociación entre una cadena preestablecida que identifica su localización (Unidad[Centro]) y la ip variable con la que es identificado en Internet en ese momento. A los efectos de asegurar el envío, se implementa un hash que permita validar el envío. Del lado del servidor se recibe la terna "localización:ip:hash", se valida y se registra en base de datos la dupla (localización, ip) Esta solución puede implementarse realizando modificaciones mínimas al sistema actual ya que solamente sustituye la identificación por puesto por la identificación por dirección IP, pudiendo implementarse en paralelo con la solución actual.

A.6.4.1. Cliente

Variables utilizadas:
loc → cadena identificadora del Centro[unidad]
ip → dirección ip variable

```
Repito cada x tiempo
hash = md5(loc+ip)
cadena = loc + ‘‘:’’ + ip + ‘‘:’’ + hash
Envío cadena
Fin repito
```

A.6.4.2. Servidor

Variables utilizadas:
loc → cadena identificadora de la localización
ip → dirección IP variable

```
Repito siempre
cadena = cadena_recibida
(loc, ip, hash) = split (‘‘:’’, cadena)
if (md5(loc+ip) == hash) {
Inserto en tabla de localizaciones el par (loc,ip)
}
Fin repito
```

A.6.4.3. Modificaciones posibles a la solución propuesta

En lugar de realizar el envío cada cierto tiempo, el cliente podría controlar variaciones en su propia dirección IP y comunicarse con el servidor para actualizarla solamente cuando es necesario. En este caso, sería necesario que la comunicación utilizara el protocolo TCP a los efectos de asegurarse que la comunicación con el servidor fue exitosa.

Costo estimado:

Desarrollo de aplicación cliente: 12 horas.

Testing de la aplicación: 4 horas.

Instalación de aplicación en un cliente: 1 hora.

Desarrollo del servicio en el servidor para recibir las notificaciones de los clientes y mapear la ubicación: 8 horas.

Modificación del servidor web para buscar una ubicación a partir de la dirección IP: 4 horas.

Testing del servidor: 8 horas.

No cuantificable: mapeo de direcciones de ip y ubicaciones en base de datos. Costo de identificar un error en la notificación y solucionarlo.

A.6.5. Certificado digital

Costo estimado:

Instalación del certificado en un cliente: 1 hora.

Modificación y configuración del servidor web: 15 horas.

Testing: 8 horas.

A.6.6. Comparación de costos

Solución	Desarrollo	Testing	Instalación en Cliente	Total para 150 clientes
#1	20 horas	4 horas	1 hora	174 horas
#2	12 horas	6 horas	1 hora	168 horas(*)
#3	8 horas	2 horas	1 hora	160 horas(**)
#4	24 horas	8 horas	1 hora	182 horas(***)
#5	15 horas	8 horas	1 hora	173 horas

Se deben considerar los siguientes costos: *: Identificar y solucionar un error en el proxy. **: Realizar mapeo de ubicaciones y direcciones IP. ***: Realizar mapeo y direcciones IP. Identificar y solucionar un error en notificación.

Anexo B

Scripts de migración

B.1. Introducción

El proceso de migración se dividió en dos etapas:

1. volcada de datos desde la base Oracle a archivos temporales
2. carga de datos desde los archivos a los nuevos motores.

B.2. Volcada de datos desde la base Oracle

La estrategia principal de la volcada es generar archivos de un tamaño relativamente pequeño, logrando unidades transaccionales cuando los datos sean cargados más adelante. El script puede generar directamente SQL o datos serializados de Python. En la práctica no se utilizó SQL debido a la intención de independizar los scripts de un motor de base de datos en particular. Fue necesario hacer algunas pequeñas correcciones de datos "on the fly", como por ejemplo corregir las fechas 1/1/1 que MySQL no soporta.

```
import sys
import os
import glob
import traceback
import time
import datetime
import pprint
from optparse import OptionParser
import cx_Oracle

def read_arguments():

    parser = OptionParser()
    parser.add_option("-u", "--unicode",
                      action="store_true", dest="unicode", default=False,
                      help="use unicode strings for cx_Oracle connection")
    parser.add_option("-t", "--tables", dest="tables",
                  help="list of comma separated tables to dump")
```

```
    parser.add_option("-f", "--file", dest="filename",
                      help="file containing tables to dump")
    parser.add_option("-l", "--limit", dest="limit",
                      type="int", default=-1,
                      help="limit number of records per table")
    parser.add_option("-p", "--page-size", dest="page_size",
                      type="int", default=100000,
                      help="number of records to store per file")
    parser.add_option("-s", "--lsql",
                      action="store_true", dest="lsql", default=False,
                      help="generate SQL statements")
    parser.add_option("-D", "--no-dat",
                      action="store_false", dest="dat", default=True,
                      help="avoid generating Python tuples")
    parser.add_option("-q", "--quiet",
                    action="store_false", dest="verbose", default=True,
                    help="don't print status messages to stdout")
    return parser.parse_args()

def log(str):
    if options.verbose:
        print str

def main():

    tables = tables_to_dump()
    oracle, oracle_cursor = connect_to_oracle()
    for table in tables:
        primary_key_list = oracle_primary_key(oracle_cursor, table)
        primary_key = ",".join(primary_key_list)
        page_number = 1
        count = 0
        while True:
            temp_files = open_temp_files(table, page_number)
            sentence = build_select_sentence(table, primary_key)
            try:
                log('=> Executing ..')
                log(sentence[:80])
                cr = oracle_cursor.execute(sentence)
                fields = [x[0] for x in cr.description]
                page_count = 0
                for row in oracle_cursor:
                    page_count += 1
                    count += 1
                    write_data_to_temps(temp_files, table, fields, row,
                        count)
                    if page_count == options.page_size:
                        rename_files(temp_files, table, page_number)
                        page_number += 1
                        page_count = 0
                        temp_files = open_temp_files(table, page_number)
                if page_count != 0:
                    rename_files(temp_files, table, page_number)
                write_descriptor_file(table, fields)
                break
            except:
                handle_error(table, page_number, count, sentence)
                sys.exit()
                break
```

```
def tables_to_dump():

    comma_separated_tables = options.tables
    if not comma_separated_tables is None and len(comma_separated_tables) >
         0:
        tables = comma_separated_tables.split(",")
    else:
        tables = []
    if len(tables) == 0:
        if not options.filename is None:
            fname = options.filename
        else:
            fname = 'tablas_sgs.txt'
        f = file(fname)
        t = f.read()
        f.close()
        tables = [x for x in t.split('\n') if x != '']
        log(pp.pprint(tables))
    return tables

def connect_to_oracle():

    log('Connecting to Oracle ..')
    connection_string = u'ssaeprod/ssaeprod@10.100.202.125:1521/ssae'
    if options.unicode:
        connection_string = unicode(connection_string)
    oracle = cx_Oracle.connect(connection_string)
    oracle_cursor = oracle.cursor()
    if options.verbose:
        print 'Connected to Oracle!'
    return oracle, oracle_cursor

def oracle_primary_key(oracle_cursor, table):

    sentence = "SELECT uc.table_name, ucc.column_name \
                FROM user_constraints uc, user_cons_columns ucc \
                    WHERE uc.constraint_name = ucc.constraint_name \
                    AND uc.constraint_type = 'P' \
                    AND INSTR(uc.table_name, 'X_') <> 1 \
                    AND uc.table_name = '%s' \
                ORDER BY uc.table_name, ucc.position" %(table.upper())
    if options.unicode:
        sentence = unicode(sentence)
    cr = oracle_cursor.execute(sentence)
    fields = []
    for c in cr:
        fields.append(c[1])
    return fields

def open_temp_files(table, page_number):
    files = {}
    if options.lsql:
        files['sql'] = open_temp_file('sql', table, page_number)
    if options.dat:
        files['dat'] = open_temp_file('dat', table, page_number)
    return files

def open_temp_file(type, table, page_number):
```

```
    filename = 'temp'
    filepath = os.path.join(type, filename)
    log('Dumping to %s' %filepath)
    file = open(filepath, 'w')
    return file

def filename(table, page_number, type):
    return '%s_%s.%s' %(table, str(page_number).zfill(5), type)

def filepath(table, page_number, type):
    return os.path.join(type, filename(table, page_number, type))

def write_data_to_temps(temp_files, table, fields, row, count):
    if options.lsql:
        write_sql_data(temp_files['sql'], table, fields, row, count)
    if options.dat:
        write_python_data(temp_files['dat'], table, fields, row, count)

def write_sql_data(file, table, fields, row, count):
    data = sql_insert_sentence(table, fields, row)
    file.write(data + '\n')
    log(' ==> count [%s]' % count)
    log(repr(data)[:80])

def write_python_data(file, table, fields, row, count):
    new_row = []
    for i in row:

        if isinstance(i, datetime.datetime) and (i.year < 1910 or i.year >
            2020) and i.year != 1:
            year_digits = str(i.year)[len(str(i.year)) - 2 : len(str(i.year
                ))]
            if len(year_digits) == 1:
                year_digits = "0" + year_digits
            if int(year_digits) > 20:
                year = int("19" + year_digits)
            else:
                year = int("20" + year_digits)
            new_row.append(datetime.datetime(year, i.month, i.day, i.hour,
                i.minute, i.second, i.microsecond))
        else:
            new_row.append(i)
    row = tuple(new_row)
    file.write(repr(row) + '\n')
    log(' ==> count [%s]' % count)
    log(repr(row)[:80])

def write_descriptor_file(table, fields):
    filepath = os.path.join('dat', '%s_descriptor' %table)
    file = open(filepath, 'w')
    data = {'table_fields': fields}
    file.write(repr(data))
    file.close()

def rename_files(temp_files, table, page_number):
    if options.lsql:
        temp_files['sql'].close()
        os.rename(temp_files['sql'].name, filepath(table, page_number, 'sql
            '))
```

```
    if options.dat:
        temp_files['dat'].close()
        src = temp_files['dat'].name
        dest = filepath(table, page_number, 'dat')
        os.rename(src, dest)
        log(dest + ' ready.')

def build_select_sentence(table, primary_key):
    sentence = "select * from %s order by %s" %(table, primary_key)
    if options.unicode:
        sentence = unicode(sentence)
    return sentence

def sql_insert_sentence(table, fields, row):

    fiel = '('
    val = '('
    new_row = []
    for i in range(0, len(row)):
        if not row[i] is None:
            val = val + '%s,'
            fiel = fiel + fields[i][0] + ','
            if isinstance(row[i], basestring):
                new_row.append("'" + row[i] + "'")
            elif isinstance(row[i], datetime.datetime):
                if row[i].year < 1000:
                    year = 1000
                else:
                    year = row[i].year
                str_date = "%s-%s-%s %s:%s:%s" %(year,
                                                 row[i].month,
                                                 row[i].day,
                                                 row[i].hour,
                                                 row[i].minute,
                                                 row[i].second)
                new_row.append("'" + str_date + "'")
            else:
                new_row.append(row[i])
    val = val[:len(val) - 1] +  ')'
    fiel = fiel[:len(fiel) - 1] + ')'
    sentence = 'insert into %s %s values %s;' %(table, fiel, val)
    sentence = sentence % tuple(new_row)
    return sentence

def handle_error(table, page_number, count, sentence):
    tr = traceback.format_exc()
    ferror = open('errors/errors.txt', 'a')
    l1 = "Error in %s" % sentence
    l2 = repr(tr)
    print l1
    print l2
    ferror.write(l1 + '\n')
    ferror.write(l2 + '\n')
    ferror.close()

pp = pprint.PrettyPrinter(indent=4)
options, args = read_arguments()

if __name__ == '__main__':
```

```
    main()
```

B.3. Carga de datos desde archivos

La carga de datos se realiza insertando todas las filas representadas en un archivo temporal y confirmando la transacción a su fin. En caso de que ocurra algún error -o una interrupción voluntaria de la carga- el archivo deberá ser procesado nuevamente por entero. Con una partición adecuada durante la etapa de volcado, se logra un proceso ágil y seguro de carga de datos al nuevo motor.

```
import os
import glob
import datetime
import time
import sys, traceback
import pprint
from optparse import OptionParser
import pgdb
import MySQLdb
import monetdb.sql

def read_arguments():

    parser = OptionParser()
    parser.add_option("-t", "--tables", dest="tables",
                  help="list of comma separated tables to load")
    parser.add_option("-f", "--file", dest="filename",
                    help="file containing tables to load")
    parser.add_option("-q", "--quiet",
                  action="store_false", dest="verbose", default=True,
                  help="don't print status messages to stdout")
    parser.add_option("-e", "--engines", dest="engines",
                    default="psql,mysql,monetdb",
                    help='psql | mysql | monetdb engines')

    return parser.parse_args()

def log(str):
    if options.verbose:
        print str

def connect_to_psql():
    psql = pgdb.connect(user='postgres', password='postgres', database='
        sgs_test')
    psql_cursor = psql.cursor()
    print 'connected to psql'
    return psql, psql_cursor

def connect_to_mysql():
    mysql = MySQLdb.connect(user='root', db='sgs_test')
    mysql_cursor = mysql.cursor()
    print 'connected to mysql'
    return mysql, mysql_cursor
```

```
def connect_to_monetdb():
    monetdb = monetdb.sql.connect(username='monetdb', password='monetdb',
        hostname='localhost', database='sgs_test')
    monetdb_cursor = monetdb.cursor()
    print 'connected to monetdb'
    return monetdb, monetdb_cursor

def process (name, db, db_cursor, sentence, tvalues):
    attempt = 1
    while True:
        try:
            success = eval('process_%s' %name)(db, db_cursor, sentence,
                tvalues)
            break
        except:
            print "Sentence: "
            print repr(sentence)
            print "Values: "
            print repr(tvalues)
            print repr(traceback.format_exc())
            print 'sleep..'
            time.sleep(30)
            attempt += 1
            print 'attempt %s' %attempt
            if attempt >= 5:
                success = False
                break
            db, db_cursor = eval('connect_%s()' %name)
    return db, db_cursor, success

def process_psql(psql, psql_cursor, sentence, values):
    try:
        sentence = sentence.encode('utf-8')
        psql_cursor.execute(sentence, values)
        psql.commit()
        return True
        print 'ok'
    except:
        tr = traceback.format_exc()
        if not 'duplicate key' in repr(tr):
            print repr(tr)
            raise Exception()
        else:
            psql_cursor.execute('rollback')
            print 'duplicate key'

def process_mysql(mysql,mysql_cursor,sentence, values):
    try:
        mysql_cursor.execute(sentence, values)
        return True
    except:
        tr = traceback.format_exc()
        if not 'Duplicate' in repr(tr):
            print repr(tr)
            raise Exception()
        else:
            print 'Duplicate Key for: %s, %s' %(sentence, repr(values))
            return False
```

```
def process_monetdb(monetdb,monetdb_cursor,sentence, values):
    try:
        monetdb_cursor.execute(sentence, values)
        return True
    except:
        tr = traceback.format_exc()
        if not 'PRIMARY KEY' in repr(tr):
            print repr(tr)
            raise Exception()
        else:
            print 'duplicate key'
            monetdb_cursor.execute('rollback')
            return False

def main():

    tables = tables_to_load()

    data_files = glob.glob('dat/*.dat')
    data_files.sort()

    if 'psql' in options.engines:
        psql, psql_cursor= connect_to_psql()
    if 'mysql' in options.engines:
        mysql, mysql_cursor = connect_to_mysql()
    if 'monetdb' in options.engines:
        monetdb, monetdb_cursor = connect_to_monetdb()

    for data_file in data_files:
        filename = data_file.split('/')[len(data_file.split('/')) - 1]
        table = filename.split('_0')[0]
        if table in tables:
            inp = open(data_file,"r")

            fields = eval(open(os.path.join('dat', table + '_descriptor'),
                'r').read())['table_fields']
            count= 0
            psql_success = True
            mysql_success = True
            monetdb_success = True
            for data_line in inp.readlines():
                count += 1
                sentence = build_insert_sentence(data_line, fields, table)
                values = []
                for i in range(0, len(e)):
                    if isinstance(e[i], datetime.datetime) and e[i].year <
                        1900:
                        values.append(datetime.datetime(1900, 1, 1, 0, 0))
                    elif isinstance(e[i], unicode):
                        values.append(e[i].encode('utf-8'))
                    else:
                        values.append(e[i])
                tvalues = tuple(values)
                if count % 1000 == 0 or count == 1:
                    print "Processsing record #%s of %s" %(count, data_file
                        )
                if 'psql' in options.engines and psql_success:
```

```
                    psql, psql_cursor, psql_success = process('psql', psql,
                         psql_cursor, sentence, tvalues)
                    if not psql_success:
                        print "Rollback"
                        psql.rollback()
                if 'mysql' in options.engines and mysql_success:
                    mysql, mysql_cursor, mysql_success = process('mysql',
                        mysql, mysql_cursor, sentence, tvalues)
                    if not mysql_success:
                        print "Rollback"
                        mysql.rollback()
                if 'monetdb' in options.engines and monetdb_success:
                    monetdb, monetdb_cursor, monetdb_success = process('
                        monetdb', monetdb, monetdb_cursor, sentence,
                        tvalues)
                     if not monetdb_success_success:
                        print "Rollback"
                        monetdb.rollback()
            if 'psql' in options.engines and psql_success:
                print "Commit .."
                psql.commit()
            if 'mysql' in options.engines and mysql_success:
                print "Commit .."
                mysql.commit()
            if 'monetdb' in options.engines and monetdb_success:
                print "Commit .."
                monetdb.commit()
            print "OK"
            inp.close()

def tables_to_load():

    comma_separated_tables = options.tables
    if not comma_separated_tables is None and len(comma_separated_tables) >
         0:
        tables = comma_separated_tables.split(",")
    else:
        tables = []
    if len(tables) == 0:
        if not options.filename is None:
            fname = options.filename
        else:
            fname = 'tablas_sgs.txt'
        f = file(fname)
        t = f.read()
        f.close()
        tables = [x for x in t.split('\n') if x != '']
    return tables

def build_insert_sentence():
    fiel = '('
    val = '('
    e = eval(data_line)
    for i in range(0, len(e)):
        val = val + '%s,'
        fiel = fiel + fields[i] + ','
    val = val[:len(val) - 1] +  ')'
    fiel = fiel[:len(fiel) - 1] + ')'
    return 'insert into ' + table + ' ' + fiel + ' values ' + val
```

```
options, args = read_arguments()
if __name__ == '__main__':
    main()
```

Anexo C

Traducción de triggers y stored procedures

C.1. Introducción

La migración de *stored procedures* y de *triggers* se realizó en la medida en que fue necesaria para permitir las pruebas de los casos de usos escogidos. Para los *stored procedures* que no fueron traducidos, se mantuvo solamente los enunciados correspondientes. En el caso de los *triggers* se encontró que la función principal es de auditoría, por lo cual en este apéndice se presentan algunos, a modo de ejemplo, a los efectos de ilustrar las principales diferencias entre Oracle y cada uno de los motores de destino.

En este apéndice se presentan las traducciones para Postgres y MySQL que fueron necesarias para lograr el funcionamiento adecuado de los casos de usos escogidos del Sistema de Gestión de Salud.

C.2. PostgreSQL

C.2.1. Stored Procedures

```
CREATE OR REPLACE FUNCTION "A1" (sesion varchar, fchini varchar,fchfin
    varchar,
usuci AuxAct.aausuci%type, centro AuxAct.aacenid%type, unidad AuxAct.
    aauniid%type,
famid AuxAct.aafamid%type, famtpo AuxAct.aafamtpo%type, subfamid AuxAct.
    aasubfamid%type,
susid AuxAct.aasusid%type, insid AuxAct.aainsid%type,
presentaid AuxAct.aapresentaid%type, estado AuxAct.aaestid%type) returns
    void AS $$
```

```
DECLARE
fchayer varchar(10);
fchhoy varchar(10);

begin
```

```
savepoint prueba;

delete from auxact where
aasesid = sesion;

commit;

if usuci = '999' then

    insert into auxact
    (AASESID,AAACTID,
    AACENID,AAUNIID,
    AAPRESENTAID,
    AAFAMID,AAFAMTPO,AASUBFAMID,AASUSID,AAINSID,
    AAINI,AAFIN,
    AAPUESTO,
    AAESTID,AATRAID,AAUSR,
    AASALDOANT, AASALDOPOST,AASALDOMOV,
    AAUSUCI,AAUSUID,AAPREID )
    SELECT                                          sesion,ActId, ActCenId,
        ActUniId, ActPresentaId, ActFamId, ActFamTpo, ActSubFamId, ActSusId
        , ActInsId, ActFchIni, ActFchFin, ' ',EstId, TraId, ActUsr,
        ActSaldoAnterior, ActSaldoPosterior, ActSaldoMovimiento, Actusuci,
        ActUsuId, PreId
    FROM ssaeprod.ACTUACION
    WHERE (ObjWId = 6 AND ActFchIni >= to_date(fchini,'DD/MM/YYYY'))
    AND (( ActFchIni >= to_date(fchini,'DD/MM/YYYY') )
    AND ( ActFchIni < to_date(fchfin,'DD/MM/YYYY') )
    AND ( ActCenId = centro )
    AND ( ActUniId = unidad )
    AND ( ActFamId = famid )
    AND ( ActFamTpo = famtpo )
    AND ( ActSubFamId = subfamid )
    AND ( ActSusId = susid )
    AND ( ActInsId = insid)
    AND ( ActPresentaId = presentaid )
    AND ( Estid = estado))
    ORDER BY ObjWId, ActFchIni;
else
    if presentaid <> 0 and centro <> 0 then
 insert into auxact
 (AASESID,AAACTID,
 AACENID,AAUNIID,
 AAPRESENTAID,
 AAFAMID,AAFAMTPO,AASUBFAMID,AASUSID,AAINSID,
 AAINI,AAFIN,
 AAPUESTO,
 AAESTID,AATRAID,AAUSR,
 AASALDOANT, AASALDOPOST,AASALDOMOV,
 AAUSUCI,AAUSUID,AAPREID )
 SELECT                                          sesion,ActId, ActCenId,
     ActUniId, ActPresentaId, ActFamId, ActFamTpo, ActSubFamId, ActSusId,
     ActInsId, ActFchIni, ActFchFin, ' ',EstId, TraId, ActUsr,
     ActSaldoAnterior, ActSaldoPosterior, ActSaldoMovimiento, ActUsuci,
     ActUsuId, PreId
 FROM ssaeprod.ACTUACION
 WHERE (ObjWId = 6 AND ActFchIni >= to_date(fchini,'DD/MM/YYYY')
 AND ( ActFchIni < to_date(fchfin,'DD/MM/YYYY') )
 AND ActUsuCi = usuci
```

```
AND ( ActCenId = centro )
AND ( ActUniId = unidad )
AND ( ActFamId = famid )
AND ( ActFamTpo = famtpo )
AND ( ActSubFamId = subfamid )
AND ( ActSusId = susid )
AND ( ActInsId = insid )
AND ( ActPresentaId = presentaid )
AND ( Estid = estado))
ORDER BY ObjWId, ActFchIni;
   else
if centro <> 0 then
    insert into auxact
    (AASESID,AAACTID,
    AACENID,AAUNIID,
    AAPRESENTAID,
    AAFAMID,AAFAMTPO,AASUBFAMID,AASUSID,AAINSID,
    AAINI,AAFIN,
    AAPUESTO,
    AAESTID,AATRAID,AAUSR,
    AASALDOANT, AASALDOPOST,AASALDOMOV,
    AAUSUCI,AAUSUID,AAPREID )
    SELECT                                          sesion,ActId, ActCenId,
        ActUniId, ActPresentaId, ActFamId, ActFamTpo, ActSubFamId,
        ActSusId, ActInsId, ActFchIni, ActFchFin, ' ',EstId, TraId, ActUsr
        , ActSaldoAnterior, ActSaldoPosterior, ActSaldoMovimiento,
        ActUsuci, ActUsuId, PreId
    FROM ssaeprod.ACTUACION
    WHERE (ObjWId = 6 AND ActFchIni >= to_date(fchini,'DD/MM/YYYY')
    AND ( ActFchIni < to_date(fchfin,'DD/MM/YYYY') )
    AND ActUsuCi = usuci
    AND ( ActCenId = centro )
    AND ( ActUniId = unidad )
    AND ( Estid = estado))
    ORDER BY ObjWId, ActFchIni;
else
    if presentaid <> 0 then
 insert into auxact
 (AASESID,AAACTID,
 AACENID,AAUNIID,
 AAPRESENTAID,
 AAFAMID,AAFAMTPO,AASUBFAMID,AASUSID,AAINSID,
 AAINI,AAFIN,
 AAPUESTO,
 AAESTID,AATRAID,AAUSR,
 AASALDOANT, AASALDOPOST,AASALDOMOV,
 AAUSUCI,AAUSUID,AAPREID )
 SELECT                                          sesion,ActId, ActCenId,
     ActUniId, ActPresentaId, ActFamId, ActFamTpo, ActSubFamId, ActSusId,
     ActInsId, ActFchIni, ActFchFin, ' ',EstId, TraId, ActUsr,
     ActSaldoAnterior, ActSaldoPosterior, ActSaldoMovimiento, ActUsuci,
     ActUsuId, PreId
 FROM ssaeprod.ACTUACION
 WHERE (ObjWId = 6 AND ActFchIni >= to_date(fchini,'DD/MM/YYYY')
 AND ( ActFchIni < to_date(fchfin,'DD/MM/YYYY') )
 AND ActUsuCi = usuci
 AND ( ActFamId = famid )
 AND ( ActFamTpo = famtpo )
 AND ( ActSubFamId = subfamid )
```

```
  AND ( ActSusId = susid )
  AND ( ActInsId = insid )
  AND ( ActPresentaId = presentaid )
  AND ( Estid = estado))
  ORDER BY ObjWId, ActFchIni;
     else
  insert into auxact
  (AASESID,AAACTID,
  AACENID,AAUNIID,
  AAPRESENTAID,
  AAFAMID,AAFAMTPO,AASUBFAMID,AASUSID,AAINSID,
  AAINI,AAFIN,
  AAPUESTO,
  AAESTID,AATRAID,AAUSR,
  AASALDOANT, AASALDOPOST,AASALDOMOV,
  AAUSUCI,AAUSUID,AAPREID )
  SELECT                                          sesion,ActId, ActCenId,
      ActUniId, ActPresentaId, ActFamId, ActFamTpo, ActSubFamId, ActSusId,
      ActInsId, ActFchIni, ActFchFin, ' ',EstId, TraId, ActUsr,
      ActSaldoAnterior, ActSaldoPosterior, ActSaldoMovimiento, ActUsuci,
      ActUsuId, PreId
  FROM ssaeprod.ACTUACION
  WHERE (ObjWId = 6 AND ActFchIni >= to_date(fchini,'DD/MM/YYYY')
  AND ( ActFchIni < to_date(fchfin,'DD/MM/YYYY') )
  AND ActUsuCi = usuci
  AND ( Estid = estado))
  ORDER BY ObjWId, ActFchIni;
     end if;
 end if;
    end if;
end if;


commit;




end;

$$ LANGUAGE plpgsql;


CREATE or replace FUNCTION "A2" (pName_user inout ssaemaeuser.name_user%
    type,
pserid inout servicio.serid%type, inout pUserValido varchar(2)) returns
    record AS $$

DECLARE


vpstrPerfil varchar(200);
cantgrantee   integer;
grole seguridad_ssae.granted_role%type;
```

```
begin

  pUserValido = 'N';
  SELECT granted_role FROM seguridad_ssae INTO grole
  WHERE  grantee = pName_user;

    select 1 into strict cantgrantee
    FROM perfser
    WHERE psserid = pserid AND nomperfil = grole;




    pUserValido = 'S';
    RETURN;

    exception

    when too_many_rows then
       pUserValido = 'S';
       raise exception 'Many rows';

    when no_data_found then
       pUserValido = 'N';
       raise exception 'No data';

end;

$$ LANGUAGE plpgsql;


CREATE OR REPLACE FUNCTION  "ACTILIZOMINIMOS" () returns void AS $$

DECLARE
fchayer varchar(10);
fchhoy varchar(10);

begin

delete from salestado where
traobjwestid = 2219;

insert into salestado
select cenid,uniid,2219,presentaid,salfamid,salfamtpo,salsubfamid,salsusid,
    salinsid,sallote,
salestactual / 2,current_date,salestactcomp
from salestado where
traobjwestid = 2222;

exception
when RAISE_EXCEPTION then
rollback;
end;

$$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE FUNCTION  "ACTUALIZOEXCEDENTE" () RETURNS void AS $$
DECLARE
v_job  integer := 99;
v_last_date  date;
v_fec  date;
v_dias integer;
v_err  varchar(2000);
begin
end;

$$ LANGUAGE plpgsql;


CREATE OR REPLACE FUNCTION   "P_AUDIT_CONSULTA1530"
(paux  varchar,
paux2 varchar,
paux3 varchar,
paux4 varchar,
paux5 varchar,
paux6 varchar) RETURNS void AS $$
begin
insert into audit_CONSULTA1530 (aud_date, consulta, consulta2, consulta3,
    consulta4, consulta5, consulta6)
values (sysdate, paux, paux2, paux3, paux4, paux5, paux6);
END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION   "P_AUDIT_CONSULTA3060"
(paux  varchar,
paux2 varchar,
paux3 varchar,
paux4 varchar,
paux5 varchar,
paux6 varchar,
paux7 varchar,
paux8 varchar,
paux9 varchar,
paux10 varchar,
paux11 varchar,
paux12 varchar) RETURNS void AS $$
begin
insert into audit_CONSULTA3060 (aud_date, consulta, consulta2, consulta3,
    consulta4, consulta5, consulta6, consulta7, consulta8, consulta9,
    consulta10, consulta11, consulta12)
values (sysdate, paux, paux2, paux3, paux4, paux5, paux6, paux7, paux8,
    paux9, paux10, paux11, paux12);
END;

$$ LANGUAGE plpgsql;


CREATE OR REPLACE FUNCTION   "P_AUDIT_EJECUTOCONSULTA"
(paux  varchar,
paux2 varchar,
paux3 varchar,
paux4 varchar,
paux5 varchar)  RETURNS void AS $$
```

```
begin
insert into audit_ejecutoconsulta (aud_date, consulta, consulta2, consulta3
    , consulta4, consulta5)
values (sysdate, paux, paux2, paux3, paux4, paux5);
END;

$$ LANGUAGE plpgsql;


CREATE OR REPLACE FUNCTION   "P_AUDIT_REPPER"
(pcenid inout centro.cenid%type,
puniid inout unidad.uniid%type,
pmovid inout movil.movid%type,
pserid inout servicio.serid%type,
phorfchddes inout horfun.horfchd%type,
phorfchdhas inout horfun.horfchd%type,
prepfch inout date,prepperint integer,
prepfunid inout funcionario.funid%type,
pboton inout varchar,
phorfunusrpre inout horfun.horfunusrpre%type) returns record AS $$


begin
insert into audit_repper (aud_date, cenid, uniid, movid, serid, horfchddes,
     horfchdhas, repfch, repperint, repfunid, boton, horfunusrpre)
values (sysdate, pcenid, puniid, pmovid, pserid, phorfchddes, phorfchdhas,
    prepfch, prepperint, prepfunid, pboton, phorfunusrpre);

END;

$$ LANGUAGE plpgsql;


CREATE OR REPLACE FUNCTION   "validuserserv"


(pName_user inout SSAEMAEUSER.Name_user%type,
pSerDsc inout servicio.serdsc%type,
pUserValido inout varchar) returns record AS $$


BEGIN

  puservalido   = 'S';

END;

$$ LANGUAGE plpgsql;
```

C.2.2. Triggers

```
CREATE OR REPLACE FUNCTION cenfun_jntrg() RETURNS TRIGGER AS
    $trigger_ejemplo$
  DECLARE operation varchar(3);
  BEGIN
       IF (TG_OP = 'INSERT') THEN
          operation = 'INS';
       ELSIF (TG_OP = 'UPDATE') THEN
          operation = 'UPD';
```

```
        ELSIF (TG_OP = 'DELETE') THEN
           operation = 'DEL';
        END IF;
        IF (TG_OP = 'UPDATE' OR TG_OP = 'INSERT') THEN
           INSERT INTO cenfun_jn (cenid, uniid, funid, cenfunfch, aud_action
               , aud_date, aud_user) VALUES (new.cenid, new.uniid, new.funid
               , new.cenfunfch, operation, current_timestamp, current_user);
        ELSE
           INSERT INTO cenfun_jn (cenid, uniid, funid, cenfunfch, aud_action
               , aud_date, aud_user) VALUES (old.cenid, old.uniid, old.funid
               , old.cenfunfch, operation, current_timestamp, current_user);
        END IF;
        RETURN new;
  END;
$trigger_ejemplo$ LANGUAGE plpgsql;

CREATE TRIGGER cenfun_jn
BEFORE INSERT OR UPDATE OR DELETE ON cenfun
    FOR EACH ROW EXECUTE PROCEDURE cenfun_jntrg();
```

C.3. MySQL

C.3.1. Stored Procedures

```
aELIMITER //

DROP PROCEDURE IF EXIST NUMERO//
CREATE PROCEDURE NUMERO (inout porigen VARCHAR(10), inout pnumero number
    (18))
begin
   insert into proclog (proc, fechora) values ('numero', now());

   if porigen = 'PREPrest1' then
      select nextval('num_prestacion') into pnumero;
   end if;
   if porigen = 'ACTUsuar4' then
      select nextval('num_actuacion_usu') into pnumero;
   end if;
   if porigen = 'ACTMovil5' then
      select nextval('num_actuacion_mov') into pnumero;
   end if;
   if porigen = 'ACTSaldo6' then
      select nextval('num_actuacion_sal') into pnumero;
   end if;
   if porigen = 'ACTPrest1' then
      select nextval('num_prestacion') into pnumero;
   end if;

end //

DROP PROCEDURE IF EXIST NEXTVAL//
CREATE PROCEDURE NEXTVAL(in seq_name varchar(100),out cur_val bigint(19))
BEGIN
    DECLARE cur_val bigint(20);

    SELECT sequence_cur_value INTO cur_val
    FROM sequence_data
    WHERE sequence_name = seq_name;
```

```
    IF cur_val IS NOT NULL THEN
        UPDATE `sgs_test`.sequence_data
        SET sequence_cur_value = IF ((sequence_cur_value +
            sequence_increment) > sequence_max_value,
            IF (sequence_cycle = TRUE, sequence_min_value,NULL),
                sequence_cur_value + sequence_increment)
        WHERE sequence_name = seq_name;
    END IF;
    RETURN cur_val;
END //


DROP PROCEDURE IF EXIST A1//
create PROCEDURE A1 (in sesion varchar(500), in fchini varchar(500),in
    fchfin varchar(500), in usuci varchar(15),
in centro int(11), in unidad int(11), in famid varchar(8), in famtpo
    varchar(3), in subfamid varchar(8),
in susid varchar(8), in insid smallint(6), in presentaid bigint(20), in
    estado smallint(6))

BEGIN
    DECLARE fchayer varchar(10);
    DECLARE fchhoy varchar(10);
    DECLARE EXIT HANDLER FOR NOT FOUND,SQLEXCEPTION
        BEGIN
            ROLLBACK TO SAVEPOINT prueba;
        END;

    INSERT INTO proclog (proc, fechora) VALUES ('a1', now());

    SAVEPOINT prueba;

    DELETE FROM auxact
    WHERE aasesid = sesion;

    COMMIT;

    IF usuci = '999' THEN
        INSERT INTO auxact(AASESID, AAACTID, AACENID, AAUNIID, AAPRESENTAID
            , AAFAMID, AAFAMTPO, AASUBFAMID, AASUSID, AAINSID,
            AAINI, AAFIN, AAPUESTO, AAESTID, AATRAID, AAUSR, AASALDOANT,
                AASALDOPOST, AASALDOMOV, AAUSUCI, AAUSUID, AAPREID)
        SELECT sesion,ActId, ActCenId, ActUniId, ActPresentaId, ActFamId,
            ActFamTpo, ActSubFamId, ActSusId, ActInsId,
            ActFchIni, ActFchFin,' ',EstId, TraId, ActUsr, ActSaldoAnterior
                , ActSaldoPosterior, ActSaldoMovimiento, Actusuci, ActUsuId
                , PreId
        FROM ACTUACION
        WHERE (ObjWId = 6 AND ActFchIni >= str_to_date(fchini,'%d/%m/%Y'))
            AND (( ActFchIni >= str_to_date(fchini,'%d/%m/%Y') )
            AND ( ActFchIni < str_to_date(fchfin,'%d/%m/%Y') ) AND (
                ActCenId = centro ) AND ( ActUniId = unidad )
            AND ( ActFamId = famid ) AND ( ActFamTpo = famtpo ) AND (
                ActSubFamId = subfamid ) AND ( ActSusId = susid )
            AND ( ActInsId = insid) AND ( ActPresentaId = presentaid ) AND
                ( Estid = estado))
        ORDER BY ObjWId, ActFchIni;

    ELSE
```

```
IF presentaid <> 0 AND centro <> 0 THEN
    INSERT INTO auxact(AASESID, AAACTID, AACENID, AAUNIID,
        AAPRESENTAID, AAFAMID, AAFAMTPO, AASUBFAMID, AASUSID,
        AAINSID,
        AAINI, AAFIN, AAPUESTO, AAESTID, AATRAID, AAUSR, AASALDOANT
            , AASALDOPOST, AASALDOMOV, AAUSUCI, AAUSUID, AAPREID )
    SELECT sesion,ActId, ActCenId, ActUniId, ActPresentaId,
        ActFamId, ActFamTpo, ActSubFamId, ActSusId, ActInsId,
       ActFchIni, ActFchFin, ' ', EstId, TraId, ActUsr,
           ActSaldoAnterior, ActSaldoPosterior, ActSaldoMovimiento,
            ActUsuci, ActUsuId, PreId
    FROM ACTUACION
    WHERE (ObjWId = 6 AND ActFchIni >= str_to_date(fchini,'%d/%m/%Y
        ') AND ( ActFchIni < str_to_date(fchfin,'%d/%m/%Y') )
        AND ActUsuCi = usuci AND ( ActCenId = centro ) AND (
            ActUniId = unidad ) AND ( ActFamId = famid )
        AND ( ActFamTpo = famtpo ) AND ( ActSubFamId = subfamid )
            AND ( ActSusId = susid ) AND ( ActInsId = insid )
        AND ( ActPresentaId = presentaid ) AND ( Estid = estado))
    ORDER BY ObjWId, ActFchIni;

ELSE
    IF centro <> 0 THEN
        INSERT INTO auxact( AASESID, AAACTID, AACENID, AAUNIID,
            AAPRESENTAID, AAFAMID, AAFAMTPO, AASUBFAMID, AASUSID,
            AAINSID,
            AAINI,AAFIN, AAPUESTO, AAESTID, AATRAID, AAUSR,
                AASALDOANT, AASALDOPOST, AASALDOMOV, AAUSUCI,
                AAUSUID, AAPREID )
        SELECT sesion,ActId, ActCenId, ActUniId, ActPresentaId,
            ActFamId, ActFamTpo, ActSubFamId, ActSusId, ActInsId,
            ActFchIni, ActFchFin, ' ',EstId, TraId, ActUsr,
                ActSaldoAnterior, ActSaldoPosterior,
                ActSaldoMovimiento, ActUsuci, ActUsuId, PreId
        FROM ACTUACION
        WHERE (ObjWId = 6 AND ActFchIni >= str_to_date(fchini,'%d/%
            m/%Y') AND ( ActFchIni < str_to_date(fchfin,'%d/%m/%Y')
             )
            AND ActUsuCi = usuci AND ( ActCenId = centro ) AND (
                ActUniId = unidad ) AND ( Estid = estado))
        ORDER BY ObjWId, ActFchIni;

    ELSE
        IF presentaid <> 0 THEN
            INSERT INTO auxact ( AASESID, AAACTID, AACENID, AAUNIID
                , AAPRESENTAID, AAFAMID, AAFAMTPO, AASUBFAMID,
                AASUSID, AAINSID, AAINI,AAFIN,
                AAPUESTO, AAESTID, AATRAID, AAUSR, AASALDOANT,
                    AASALDOPOST, AASALDOMOV, AAUSUCI, AAUSUID,
                    AAPREID )
            SELECT  sesion,ActId, ActCenId, ActUniId, ActPresentaId
                , ActFamId, ActFamTpo, ActSubFamId, ActSusId,
                ActInsId, ActFchIni, ActFchFin,
                ' ',EstId, TraId, ActUsr, ActSaldoAnterior,
                    ActSaldoPosterior, ActSaldoMovimiento, ActUsuci
                    , ActUsuId, PreId
            FROM ACTUACION
            WHERE (ObjWId = 6 AND ActFchIni >= str_to_date(fchini,'
                %d/%m/%Y') AND ( ActFchIni < str_to_date(fchfin,'%d
```

```
                        /%m/%Y') )
                        AND ActUsuCi = usuci AND ( ActFamId = famid ) AND (
                             ActFamTpo = famtpo ) AND ( ActSubFamId =
                            subfamid )
                        AND ( ActSusId = susid ) AND ( ActInsId = insid )
                            AND ( ActPresentaId = presentaid ) AND ( Estid
                            = estado))
                    ORDER BY ObjWId, ActFchIni;

                ELSE
                    INSERT INTO auxact (AASESID, AAACTID, AACENID, AAUNIID,
                         AAPRESENTAID, AAFAMID, AAFAMTPO, AASUBFAMID,
                        AASUSID, AAINSID,
                        AAINI, AAFIN, AAPUESTO, AAESTID, AATRAID, AAUSR,
                            AASALDOANT, AASALDOPOST, AASALDOMOV, AAUSUCI,
                            AAUSUID, AAPREID )
                    SELECT sesion,ActId, ActCenId, ActUniId, ActPresentaId,
                         ActFamId, ActFamTpo, ActSubFamId, ActSusId,
                        ActInsId,
                        ActFchIni, ActFchFin, ' ',EstId, TraId, ActUsr,
                            ActSaldoAnterior, ActSaldoPosterior,
                            ActSaldoMovimiento, ActUsuci, ActUsuId, PreId
                    FROM ACTUACION
                    WHERE (ObjWId = 6 AND ActFchIni >= str_to_date(fchini,'
                        %d/%m/%Y') AND ( ActFchIni < str_to_date(fchfin,'%d
                        /%m/%Y') )
                        AND ActUsuCi = usuci AND ( Estid = estado))
                    ORDER BY ObjWId, ActFchIni;
                END IF;
            END IF;
        END IF;
    END IF;
    COMMIT;
END;
//

CREATE PROCEDURE A2(inout pName_user varchar(200),inout pSerId int(11),
    inout pUserValido varchar(1))
BEGIN

    DECLARE vpstrPerfil VARCHAR(250);
    DECLARE cantgrantee INT DEFAULT 0;
    DECLARE no_data INT DEFAULT 0;

    DECLARE CGRANTEE CURSOR FOR
        SELECT GRANTED_ROLE
        FROM seguridad_SSAE
        WHERE GRANTEE = pName_user;

    SET pUserValido   = 'N';

    INSERT INTO proclog (proc, fechora) VALUES ('A2', now());

    OPEN CGRANTEE;
    BEGIN
        DECLARE CONTINUE HANDLER FOR NOT FOUND set no_data=1;
        FETCH CGRANTEE INTO vpstrPerfil;
    END;
```

```
    WHILE (no_data = 0) DO

        SELECT count(*) INTO cantgrantee
        FROM PERFSER
        WHERE PSSERID = pSerId AND NOMPERFIL = vpstrPerfil;

        IF cantgrantee > 0 THEN
            SET pUserValido='S';
            SET no_data=2;
        END IF;

        BEGIN
            DECLARE CONTINUE HANDLER FOR NOT FOUND set no_data=1;
            FETCH CGRANTEE INTO vpstrPerfil;
        END;

    END WHILE;
    CLOSE CGRANTEE;
END;
//

CREATE PROCEDURE ACTILIZOMINIMOS ()
BEGIN
    DECLARE CONTINUE HANDLER FOR SQLWARNING,NOT FOUND,SQLEXCEPTION ROLLBACK
        ;
    DELETE FROM salestado
        WHERE traobjwestid = 2219;
    INSERT INTO salestado
        SELECT cenid,uniid,2219,presentaid,salfamid,salfamtpo,salsubfamid,
            salsusid,salinsid,sallote,
            salestactual / 2, CURRENT_TIMESTAMP, salestactcomp
        FROM salestado WHERE traobjwestid = 2222;
    COMMIT;
END;
//


CREATE PROCEDURE ACTUALIZOEXCEDENTE()
BEGIN
    DECLARE v_job INT DEFAULT 99;
    DECLARE v_last_date DATE;
    DECLARE v_fec DATE;
    DECLARE v_dias INT;
    DECLARE v_err VARCHAR(2000);

   DECLARE c_ult_fec CURSOR FOR SELECT parvaldte FROM paramdefault WHERE
       parid = 'JOBEXCE';
    DECLARE EXIT HANDLER FOR NOT FOUND,SQLEXCEPTION
         BEGIN

            ROLLBACK;

            SET v_err = sqlerrm;
            INSERT INTO error_jobs (id, fecha, texto) VALUES (v_job,
                CURRENT_TIMESTAMP, 'Se ha generado una excepcion al
                actualizar los datos');
            COMMIT;
            CLOSE c_ult_fec;
```

```
        END;

   SET v_fec = CURRENT_TIMESTAMP;


    OPEN c_ult_fec;
        FETCH c_ult_fec INTO v_last_date;


        SET v_dias = curdate() - IFNULL(v_last_date,curdate()-1);


        BEGIN
            IF v_dias < 1 THEN
                INSERT INTO error_jobs (id, fecha, texto) VALUES (v_job,
                    curdate(), 'No puede correr dos veces el mismo dia');
            ELSE

                UPDATE preins SET pin_cronicoed = 0, pin_cronicoexced = 0
                WHERE pin_fchproxretiro <= v_fec AND pin_estid IN
                    (1224,1225) AND pin_cronicoed < pin_cantdosis * v_dias
                AND pin_cronicoed > 0;

                UPDATE preins SET pin_cronicoed = pin_cronicoed -
                    pin_cantdosis * v_dias, pin_cronicoexced = truncate(
                    pin_cronicoed - pin_cantdosis*v_dias,0)
                WHERE pin_fchproxretiro <= v_fec AND pin_estid IN
                    (1224,1225) AND pin_cronicoed >= pin_cantdosis * v_dias
                    ;

                UPDATE paramdefault SET parvaldte = curdate() WHERE parid =
                     'JOBEXCE';

                COMMIT;
            END IF;
        END;
   CLOSE c_ult_fec;
END;
//

CREATE PROCEDURE ACTUALIZOPAUTA (in Fchini varchar(30), in Fchfin varchar
    (30))
BEGIN

    DECLARE v_CANTDOSIS DECIMAL(6,2);
    DECLARE v_CANTUSADA DECIMAL(6,2);
    DECLARE v_CRONICOED DECIMAL(13,3);
    DECLARE v_CRONICOEXCED BIGINT(20);

    DECLARE v_FCH_DESPACHO   TIMESTAMP;
    DECLARE v_CANT_DESPACHO  BIGINT(20);

    DECLARE v_dias INT;

    DECLARE consulta varchar(1000);


    DECLARE v_fchVeinticinco  varchar(15);
    DECLARE v_fchVeinticinco2 varchar(15);
```

```
DECLARE v_serv_cronico  INT(11);
DECLARE v_pauta BIGINT;


DECLARE v_preid BIGINT(20);
DECLARE v_pin_usuci VARCHAR(15) ;
DECLARE v_pin_usuid INT(11);
DECLARE v_pin_famid  VARCHAR(8);
DECLARE v_pin_famtpo VARCHAR(3);
DECLARE v_pin_subfamid  VARCHAR(8);
DECLARE v_pin_susid VARCHAR(8);
DECLARE v_pin_insid SMALLINT;
DECLARE v_pin_fecha DATE;
DECLARE v_pin_cantdsp BIGINT(20);
DECLARE v_pin_cantins BIGINT(20);
DECLARE v_presentaid BIGINT(20);
DECLARE v_pin_id SMALLINT(6);


DECLARE  c_despachos CURSOR FOR
SELECT PREID, PRESENTAID, PIN_FAMID, PIN_FAMTPO, PIN_SUBFAMID,
    PIN_SUSID, PIN_INSID,  PIN_ID, PIN_USUCI, PIN_USUID, PIN_FECHA,
    PIN_CANTDSP, PIN_CANTINS
FROM preins
WHERE pin_estid = 2227 AND (pin_cronico = 1 OR pin_cronico IS NULL) AND
     pin_fecha >= str_to_date(v_fchVeinticinco ,'%d/%m/%Y')
    AND pin_fecha < str_to_date(v_fchVeinticinco2,'%d/%m/%Y');

DECLARE  c_linea CURSOR FOR
SELECT PIN_CANTDOSIS, PIN_CRONICOEXCED, PIN_CRONICOED
FROM preins
WHERE PIN_USUCI = v_pin_usuci AND PIN_USUID = v_pin_usuid AND (
    pin_estid = 1224 OR pin_estid = 1225) AND PRESENTAID = 0 AND
    PIN_FAMID = v_pin_famid
    AND PIN_FAMTPO =  v_pin_famtpo AND PIN_SUBFAMID = v_pin_subfamid
        AND PIN_SUSID = v_pin_susid AND PIN_INSID = v_pin_insid;


SET v_fchVeinticinco=  Fchini;
SET v_fchVeinticinco2=  Fchfin;

OPEN c_despachos;

    loop_despachos: LOOP

        FETCH c_despachos INTO  v_preid, v_presentaid, v_pin_famid,
            v_pin_famtpo, v_pin_subfamid, v_pin_susid, v_pin_insid,
            v_pin_id, v_pin_usuci, v_pin_usuid,
            v_pin_fecha, v_pin_cantdsp, v_pin_cantins;

        SET v_FCH_DESPACHO   = v_pin_fecha;
        SET v_CANT_DESPACHO = v_pin_cantins;



        SET v_dias=  DATEDIFF(CURDATE(),v_FCH_DESPACHO);
```

```
            OPEN c_linea;

                FETCH c_linea INTO v_CANTDOSIS , v_CRONICOEXCED ,
                    v_CRONICOED;

                SET v_CANTUSADA= v_CANTDOSIS * v_dias;

                IF v_CANTUSADA >= v_CRONICOED THEN
                    SET v_CRONICOEXCED = 0;
                    SET v_CRONICOED = 0;
                ELSE
                    SET v_CRONICOEXCED = v_CRONICOEXCED - truncate(
                        v_CANTUSADA,0);
                    SET v_CRONICOED = v_CRONICOED - v_CANTUSADA;
                END IF;

                UPDATE preins
                SET PIN_CRONICOEXCED = v_CRONICOEXCED, PIN_CRONICOED =
                    v_CRONICOED
                WHERE PREID=c_preid AND  PRESENTAID=v_presentaid AND
                    PIN_FAMID=v_pin_famid AND PIN_FAMTPO= v_pin_famtpo AND
                    PIN_SUBFAMID=v_pin_subfamid AND
                    PIN_SUSID=v_pin_susid AND  PIN_INSID=v_pin_insid AND
                        PIN_ID=v_pin_id;

            CLOSE c_linea;

        END LOOP loop_despachos;

    CLOSE c_despachos;

END;

CREATE PROCEDURE ACTUALIZOPAUTA2 ()
BEGIN

    DECLARE v_CANTDOSIS DECIMAL(6,2);
    DECLARE v_CANTUSADA DECIMAL(6,2);
    DECLARE v_CRONICOED DECIMAL(13,3);
    DECLARE v_CRONICOEXCED BIGINT(20);

    DECLARE v_FCH_DESPACHO   TIMESTAMP;
    DECLARE v_CANT_DESPACHO  BIGINT(20);

    DECLARE v_dias INT;

    DECLARE consulta varchar(1000);


    DECLARE v_fchVeinticinco  varchar(15);
    DECLARE v_fchVeinticinco2 varchar(15);

    DECLARE v_serv_cronico  INT(11);
    DECLARE v_pauta BIGINT;


    DECLARE v_preid BIGINT(20);
    DECLARE v_pin_usuci VARCHAR(15) ;
    DECLARE v_pin_usuid INT(11);
```

```
DECLARE v_pin_famid  VARCHAR(8);
DECLARE v_pin_famtpo VARCHAR(3);
DECLARE v_pin_subfamid  VARCHAR(8);
DECLARE v_pin_susid VARCHAR(8);
DECLARE v_pin_insid SMALLINT;
DECLARE v_pin_fecha DATE;
DECLARE v_pin_cantdsp BIGINT(20);
DECLARE v_pin_cantins BIGINT(20);
DECLARE v_presentaid BIGINT(20);
DECLARE v_pin_id SMALLINT(6);

DECLARE  c_despachos CURSOR FOR
SELECT PREID, PRESENTAID, PIN_FAMID, PIN_FAMTPO, PIN_SUBFAMID,
    PIN_SUSID, PIN_INSID,  PIN_ID, PIN_USUCI, PIN_USUID, PIN_FECHA,
    PIN_CANTDSP, PIN_CANTINS
FROM preins
WHERE PIN_ESTID IN (2215,2221) AND (PIN_CRONICO = 1 OR PIN_CRONICO IS
    NULL) AND PIN_FECHA >= str_to_date(v_fchVeinticinco ,'%d/%m/%Y')
    AND PIN_FECHA < str_to_date(v_fchVeinticinco2,'%d/%m/%Y');

DECLARE  c_linea CURSOR FOR
SELECT PIN_CANTDOSIS, PIN_CRONICOEXCED, PIN_CRONICOED
FROM preins
WHERE PIN_USUCI = v_pin_usuci AND PIN_USUID = v_pin_usuid AND pin_estid
     = 1224  AND PRESENTAID = 0 AND PIN_FAMID = v_pin_famid
    AND PIN_FAMTPO =  v_pin_famtpo AND PIN_SUBFAMID = v_pin_subfamid
        AND PIN_SUSID = v_pin_susid AND PIN_INSID = v_pin_insid;


SET v_fchVeinticinco   = DATE_FORMAT((CURDATE() - 25),'%d/%m%/%Y');
SET v_fchVeinticinco2 =  DATE_FORMAT((CURDATE() - 24),'%d/%m%/%Y');

OPEN c_despachos;

    loop_despachos: LOOP

        FETCH c_despachos INTO  v_preid, v_presentaid, v_pin_famid,
            v_pin_famtpo, v_pin_subfamid, v_pin_susid, v_pin_insid,
            v_pin_id, v_pin_usuci, v_pin_usuid,
            v_pin_fecha, v_pin_cantdsp, v_pin_cantins;

        SET v_FCH_DESPACHO   = v_pin_fecha;
        SET v_CANT_DESPACHO = v_pin_cantins;

        SET v_dias=  DATEDIFF(CURDATE(),v_FCH_DESPACHO);

        OPEN c_linea;

            FETCH c_linea INTO v_CANTDOSIS , v_CRONICOEXCED ,
                v_CRONICOED;

            SET v_CANTUSADA= v_CANTDOSIS * v_dias;

            IF v_CANTUSADA >= v_CRONICOED THEN
                SET v_CRONICOEXCED = 0;
                SET v_CRONICOED = 0;
            ELSE
                SET v_CRONICOEXCED = v_CRONICOEXCED - TRUNCATE(
                    v_CANTUSADA,0);
```

```
                    SET v_CRONICOED = v_CRONICOED - v_CANTUSADA;
                END IF;

                UPDATE preins
                SET PIN_CRONICOEXCED = v_CRONICOEXCED, PIN_CRONICOED =
                    v_CRONICOED
                WHERE PREID=v_preid AND  PRESENTAID=v_presentaid AND
                    PIN_FAMID=v_pin_famid AND PIN_FAMTPO= v_pin_famtpo AND
                    PIN_SUBFAMID=v_pin_subfamid AND
                    PIN_SUSID=v_pin_susid AND  PIN_INSID=v_pin_insid AND
                        PIN_ID=v_pin_id;

            CLOSE c_linea;

        END LOOP loop_despachos;

    CLOSE c_despachos;

END;



CREATE PROCEDURE ACTUALIZOPAUTA3 ()
BEGIN

    DECLARE v_fchVeinticinco VARCHAR(10);
    DECLARE v_fchVeinticinco2 VARCHAR(10);

    DECLARE v_preid BIGINT(20);
    DECLARE v_pin_usuci VARCHAR(15) ;
    DECLARE v_pin_usuid INT(11);
    DECLARE v_pin_famid  VARCHAR(8);
    DECLARE v_pin_famtpo VARCHAR(3);
    DECLARE v_pin_subfamid  VARCHAR(8);
    DECLARE v_pin_susid VARCHAR(8);
    DECLARE v_pin_insid SMALLINT;
    DECLARE v_pin_fecha DATE;
    DECLARE v_pin_cantdsp BIGINT(20);
    DECLARE v_pin_cantins BIGINT(20);
    DECLARE v_presentaid BIGINT(20);
    DECLARE v_pin_id SMALLINT(6);
    DECLARE v_pin_cronicoexced bigint;
    DECLARE v_pin_cronicoed  NUMERIC(13,3);
    DECLARE v_dias INT;
    DECLARE  c_despachos CURSOR FOR
    SELECT PREID, PRESENTAID, PIN_FAMID, PIN_FAMTPO, PIN_SUBFAMID,
        PIN_SUSID, PIN_INSID,  PIN_ID, PIN_USUCI, PIN_USUID, PIN_FECHA,
        PIN_CANTDSP, PIN_CANTINS
    FROM preins
    WHERE PIN_ESTID IN (2215,2221) AND (PIN_CRONICO = 1 OR PIN_CRONICO IS
        NULL) AND PIN_FECHA >= STR_TO_DATE(v_fchVeinticinco ,'%d/%m/%Y')
        AND PIN_FECHA < STR_TO_DATE(v_fchVeinticinco2,'%d/%m/%Y');

    SET v_fchVeinticinco   =  DATE_FORMAT(CURDATE() - 25,'%d/%m/%Y');
    SET v_fchVeinticinco2  =  DATE_FORMAT(CURDATE() - 24,'%d/%m/%Y');

    OPEN c_despachos;
```

```
        loop_despachos: LOOP

            FETCH c_despachos INTO  v_preid, v_presentaid, v_pin_famid,
                v_pin_famtpo, v_pin_subfamid, v_pin_susid, v_pin_insid,
                v_pin_id, v_pin_usuci, v_pin_usuid,
                v_pin_fecha, v_pin_cantdsp, v_pin_cantins;

            SET v_dias=  DATEDIFF(CURDATE(),v_pin_fecha);

            IF (v_pin_cantins>= v_pin_cronicoed) THEN
                SET v_pin_cronicoexced=0;
            ELSE
                SET v_pin_cronicoexced= v_pin_cronicoexced - TRUNCATE(
                    v_pin_cantins,0);
            END IF;

            IF (v_pin_cantins >= v_pin_cronicoed) THEN
                SET v_pin_cronicoed=0;
            ELSE
                SET v_pin_cronicoed= v_pin_cronicoed - v_pin_cantins;
            END IF;

            UPDATE preins SET PIN_CRONICOEXCED=v_pin_cronicoexced,
                PIN_CRONICOED=v_pin_cronicoed
            where PIN_USUCI = v_pin_usuci and PIN_USUID = v_pin_usuid and
                pin_estid = 1224 and PRESENTAID = 0 and PIN_FAMID =
                v_pin_famid and PIN_FAMTPO =  v_pin_famtpo
                and PIN_SUBFAMID = v_pin_subfamid and PIN_SUSID =
                    v_pin_susid and PIN_INSID = v_pin_insid and
                    PIN_FCHPROXRETIRO < (CURDATE() + 5);

            COMMIT;

        END LOOP;
    CLOSE c_despachos;

END;


CREATE PROCEDURE  ACTUALIZOPAUTA4(IN Fchini VARCHAR(30), IN Fchfin VARCHAR
    (30))
BEGIN

    DECLARE v_fchVeinticinco VARCHAR(10);
    DECLARE v_fchVeinticinco2 VARCHAR(10);

    DECLARE v_preid BIGINT(20);
    DECLARE v_pin_usuci VARCHAR(15) ;
    DECLARE v_pin_usuid INT(11);
    DECLARE v_pin_famid  VARCHAR(8);
    DECLARE v_pin_famtpo VARCHAR(3);
    DECLARE v_pin_subfamid  VARCHAR(8);
    DECLARE v_pin_susid VARCHAR(8);
    DECLARE v_pin_insid SMALLINT;
    DECLARE v_pin_fecha DATE;
    DECLARE v_pin_cantdsp BIGINT(20);
    DECLARE v_pin_cantins BIGINT(20);
    DECLARE v_presentaid BIGINT(20);
    DECLARE v_pin_id SMALLINT(6);
```

```
DECLARE v_pin_cronicoexced bigint;
DECLARE v_pin_cronicoed  numeric(13,3);
DECLARE v_pin_cantdosis numeric(6,2);

DECLARE v_dias INT;
DECLARE consulta VARCHAR(1000);

DECLARE  c_despachos CURSOR FOR
SELECT PREID, PRESENTAID, PIN_FAMID, PIN_FAMTPO, PIN_SUBFAMID,
    PIN_SUSID, PIN_INSID,  PIN_ID, PIN_USUCI, PIN_USUID, PIN_FECHA,
    PIN_CANTDSP, PIN_CANTINS, PIN_CANTDOSIS
FROM preins
WHERE PIN_ESTID=2227 AND (PIN_CRONICO = 1 OR PIN_CRONICO IS NULL) AND
    PIN_FECHA >= STR_TO_DATE(v_fchVeinticinco ,'%d/%m/%Y')
    AND PIN_FECHA < STR_TO_DATE(v_fchVeinticinco2,'%d/%m/%Y');

SET v_fchVeinticinco=  Fchini;
SET v_fchVeinticinco2=  Fchfin ;

OPEN c_despachos;

    loop_despachos: LOOP

        FETCH c_despachos INTO  v_preid, v_presentaid, v_pin_famid,
            v_pin_famtpo, v_pin_subfamid, v_pin_susid, v_pin_insid,
            v_pin_id, v_pin_usuci, v_pin_usuid,
            v_pin_fecha, v_pin_cantdsp, v_pin_cantins, v_pin_cantdosis;

        SET v_dias=  DATEDIFF(CURDATE(),v_pin_fecha);



        IF (v_pin_cantdosis * v_dias >= v_pin_cronicoed) THEN
            SET v_pin_cronicoexced=0;
        ELSE
            SET v_pin_cronicoexced= v_pin_cronicoexced - TRUNCATE(
                v_pin_cantdosis * v_dias,0);
        END IF;

        IF (v_pin_cantdosis * v_dias >= v_pin_cronicoed) THEN
            SET v_pin_cronicoed=0;
        ELSE
            SET v_pin_cronicoed= v_pin_cronicoed - v_pin_cantdosis*
                v_dias;
        END IF;

        UPDATE preins SET PIN_CRONICOEXCED=v_pin_cronicoexced,
            PIN_CRONICOED=v_pin_cronicoed
        where PIN_USUCI = v_pin_usuci and PIN_USUID = v_pin_usuid and
            pin_estid = 1224 and PRESENTAID = 0 and PIN_FAMID =
            v_pin_famid and PIN_FAMTPO =  v_pin_famtpo
            and PIN_SUBFAMID = v_pin_subfamid and PIN_SUSID =
                v_pin_susid and PIN_INSID = v_pin_insid and
                PIN_FCHPROXRETIRO < CURDATE();

        COMMIT;

    END LOOP;
```

```
    CLOSE c_despachos;
END;



CREATE PROCEDURE ACTUALIZOPRECIO ()
BEGIN
    DECLARE fchayer VARCHAR(10);
    DECLARE fchhoy VARCHAR(10);
    DECLARE EXIT HANDLER FOR NOT FOUND,SQLEXCEPTION
   begin
      ROLLBACK;
   end;


    SET fchayer = DATE_FORMAT(curdate()-1,'%d/%m/%Y');
    SET fchhoy  = DATE_FORMAT(curdate(),'%d/%m/%Y');
```



```
    UPDATE preins
    SET pin_preciocompra = pin_precio * pin_cantrec
    WHERE pin_Estid = 2003 AND pin_fecha >= STR_TO_DATE(fchayer,'%d/%m/%Y')
         AND pin_Fecha < STR_TO_DATE(fchhoy,'%d/%m/%Y');
    commit;


    savepoint prueba;

    update preins
    set pin_precio = (select b.apprecio
                  from auxprecio b
                  where b.appresentaid = preins.presentaid and apprecio <>
```

```
                    0)
where pin_fecha >= str_to_date(fchayer,'%d/%m/%Y') and pin_fecha <
    str_to_date(fchhoy,'%d/%m/%Y')
    and pin_estid in (2213,2216,2214,2224,2225,2228,2229,2002);


update preins
set pin_precio = pin_precio * pin_cantdsp
where pin_estid = 2216 and pin_fecha >= str_to_date(fchayer,'%d/%m/%Y')
     and pin_fecha < str_to_date(fchhoy,'%d/%m/%Y') and pin_cantdsp <>
    0;


update preins
set pin_precio = pin_precio * pin_cantexp
where pin_estid = 2213 and pin_fecha >= str_to_date(fchayer,'%d/%m/%Y')
     and pin_fecha < str_to_date(fchhoy,'%d/%m/%Y') and pin_cantexp <>
    0;

commit;


update preins
set pin_precio = pin_precio * PIN_CANTREC
where pin_estid = 2214 and pin_fecha >= str_to_date(fchayer,'%d/%m/%Y')
     and pin_fecha < str_to_date(fchhoy,'%d/%m/%Y')
    and PIN_CANTREC <> 0;


update preins
set pin_precio = pin_precio * PIN_CANTREC
where pin_estid = 2224 and pin_fecha >= str_to_date(fchayer,'%d/%m/%Y')
     and pin_fecha < str_to_date(fchhoy,'%d/%m/%Y')
    and PIN_CANTREC <> 0;


update preins
set pin_precio = pin_precio * PIN_CANTREC
where pin_estid = 2225 and pin_fecha >= str_to_date(fchayer,'%d/%m/%Y')
     and pin_fecha < str_to_date(fchhoy,'%d/%m/%Y')
    and PIN_CANTREC <> 0;


update preins
set pin_precio = pin_precio * PIN_CANTNOREC
where pin_estid = 2228 and pin_fecha >= str_to_date(fchayer,'%d/%m/%Y')
     and pin_fecha < str_to_date(fchhoy,'%d/%m/%Y')
    and PIN_CANTNOREC <> 0;


update preins
set pin_precio = pin_precio * PIN_CANTINSFC
where pin_estid = 2229 and pin_fecha >= str_to_date(fchayer,'%d/%m/%Y')
     and pin_fecha < str_to_date(fchhoy,'%d/%m/%Y')
    and PIN_CANTINSFC <> 0;


update preins
set pin_precio = pin_precio * PIN_CANTIDAD
```

```
    where pin_estid = 2002 and pin_fecha >= str_to_date(fchayer,'%d/%m/%Y')
         and pin_fecha < str_to_date(fchhoy,'%d/%m/%Y')
        and PIN_CANTIDAD <> 0;

    commit;
END;



CREATE PROCEDURE ACTUALIZOPREINS ()
BEGIN

    DECLARE fchDiez VARCHAR(10);
    DECLARE fchDiez2 VARCHAR(10);
    DECLARE fchVeinticinco VARCHAR(10);
    DECLARE fchVeinticinco2 VARCHAR(10);
    DECLARE fchTreinta VARCHAR(10);
    DECLARE fchTreinta2 VARCHAR(10);
    DECLARE fchSesenta VARCHAR(10);
    DECLARE fchSesenta2 VARCHAR(10);
    DECLARE EXIT HANDLER FOR NOT FOUND,SQLEXCEPTION
        begin
            rollback;
        end;

    SET fchDiez  = DATE_FORMAT(curdate()-10,'%d/%m/%Y');
    SET fchDiez2 = DATE_FORMAT(curdate()-9,'%d/%m/%Y');

    SET fchVeinticinco  = DATE_FORMAT(curdate()-25,'%d/%m/%Y');
    SET fchVeinticinco2 = DATE_FORMAT(curdate()-24,'%d/%m/%Y');

    SET fchTreinta  = DATE_FORMAT(curdate()-30,'%d/%m/%Y');
    SET fchTreinta2 = DATE_FORMAT(curdate()-29,'%d/%m/%Y');

    SET fchSesenta  = DATE_FORMAT(curdate()-60,'%d/%m/%Y');
    SET fchSesenta2 = DATE_FORMAT(curdate()-59,'%d/%m/%Y');


    SAVEPOINT prueba;

    UPDATE preins
    SET pin_estid = 2227
    WHERE pin_estid IN (2215,2221) AND pin_cronico = 0 AND pin_fecha >=
        str_to_date(fchDiez,'%d/%m/%Y') AND pin_fecha < str_to_date(
        fchDiez2,'%d/%m/%Y');

    UPDATE preins
    SET pin_estid = 2227
    WHERE pin_estid IN (2215,2221) AND (pin_cronico = 1 OR pin_cronico IS
        NULL) AND pin_fecha >= str_to_date(fchVeinticinco ,'%d/%m/%Y')
        AND pin_fecha < str_to_date(fchVeinticinco2,'%d/%m/%Y');



    UPDATE preins
    SET pin_estid = 1225, pin_cronicoExced = 0
    WHERE pin_estid = 1224 AND pin_fchvigpauta >= str_to_date(fchTreinta,'%
        d/%m/%Y') AND pin_fchvigpauta < str_to_date(fchTreinta2,'%d/%m/%Y')
        ;
```

```
    COMMIT;

END;

CREATE PROCEDURE ACTUALIZOSUMINISTRO ()
BEGIN

    DECLARE fchayer VARCHAR(10);
    DECLARE fchhoy VARCHAR(10);
    DECLARE EXIT HANDLER FOR NOT FOUND,SQLEXCEPTION
        BEGIN
            ROLLBACK;
        END;


    SAVEPOINT prueba;

    SET fchayer= DATE_FORMAT(curdate()-1,"%d/%m/%Y");
    SET fchhoy= DATE_FORMAT(curdate(),"%d/%m/%Y");

    UPDATE preins
    SET PIN_CRONICO = 2
    WHERE PIN_FECHA >= str_to_date(fchayer,'%d/%m/%Y') AND PIN_FECHA <
        str_to_date(fchhoy,'%d/%m/%Y') AND PIN_SERID = 805 AND PIN_CRONICO
        <> 2;

    UPDATE preins
    SET PIN_CRONICO = 2
    WHERE PIN_FECHA >= str_to_date(fchayer,'%d/%m/%Y') and PIN_FECHA <
        str_to_date(fchhoy,'%d/%m/%Y') AND PIN_SERID = 805 AND PIN_CRONICO
        IS NULL;

    COMMIT;
END;

CREATE PROCEDURE AUXACTUACION (in sesion varchar(500), in fchini varchar
    (500),in fchfin varchar(500),
    in centro int(11), in famid varchar(8), in famtpo varchar(3), in
        subfamid varchar(8), in susid varchar(8),
    in insid smallint(6), in presentaid bigint(20), in estado smallint(6))
BEGIN

    DECLARE EXIT HANDLER FOR NOT FOUND,SQLEXCEPTION
   begin
      rollback;
   end;

    savepoint prueba;

    delete
    from auxact
    where aasesid = sesion;

    commit;
```

```
    insert into auxact (AASESID, AAACTID, AACENID, AAUNIID, AAPRESENTAID ,
        AAFAMID,AAFAMTPO , AASUBFAMID , AASUSID, AAINSID,
        AAINI, AAFIN, AAPUESTO , AAESTID, AATRAID, AAUSR, AASALDOANT ,
            AASALDOPOST, AASALDOMOV , AAUSUCI,AAUSUID, AAPREID)
    SELECT  sesion,ActId, ActCenId, ActUniId, ActPresentaId, ActFamId,
        ActFamTpo, ActSubFamId, ActSusId, ActInsId,
        ActFchIni, ActFchFin, ' ',EstId, TraId, ActUsr, ActSaldoAnterior,
            ActSaldoPosterior, ActSaldoMovimiento, ActUsuCI, ActUsuId,
            PreId
    FROM ssaeprod.ACTUACION
    WHERE (ObjWId = 6 AND ActFchIni >= str_to_date(fchini,'%d/%m/%Y')) AND
        (( ActFchIni >= str_to_date(fchini,'%d/%m/%Y') ) AND ( ActFchIni <
        str_to_date(fchfin,'%d/%m/%Y') )
        AND ( ActCenId = centro ) AND ( ActUniId >= 0 ) AND ( ActFamId =
            famid ) AND ( ActFamTpo = famtpo ) AND ( ActSubFamId = subfamid
             ) AND ( ActSusId = susid )
        AND ( ActInsId = insid) AND ( ActPresentaId = presentaid ) AND (
            Estid = estado))
    ORDER BY ObjWId, ActFchIni;

    commit;

END;


create procedure            BORROGERENCIAL ()

begin
end;



create procedure            BORROMIRRORACTUACION ()

begin
end;



CREATE PROCEDURE CALCULOFECHA()
BEGIN

    DECLARE fchMin varchar(10);
    DECLARE fchMax varchar(10);
    DECLARE EXIT HANDLER FOR NOT FOUND,SQLEXCEPTION
        BEGIN
            ROLLBACK;
        END;


    SET fchMin= DATE_FORMAT(curdate()-1,"%d/%m/%Y");
    SET fchMax= DATE_FORMAT(curdate(),"%d/%m/%Y");



    SAVEPOINT prueba;

    UPDATE prestacion
    set preaa = CAST(DATE_FORMAT(prehorfchd,'%Y') AS UNSIGNED ), premm =
```

```
        CAST(DATE_FORMAT(prehorfchd,'%m') AS UNSIGNED )
    WHERE prehorfchd >= STR_TO_DATE(fchmin,'%d/%m/%Y') AND prehorfchd <
        STR_TO_DATE(fchmax,'%d/%m/%Y')
        AND preserid < 700;

    COMMIT;

END;


DROP PROCEDURE IF EXIST CAMBIOCI//
CREATE PROCEDURE CAMBIOCI (in ciOld varchar(15), in ciidOld int(11), in
    ciNew varchar(15), in ciidNew int(11))

BEGIN
   DECLARE cntusu INT;
   DECLARE EXIT HANDLER FOR NOT FOUND,SQLEXCEPTION
       begin
          rollback to savepoint prueba;
       end;

   INSERT INTO proclog (proc, fechora) VALUES ('CAMBIOCI', now());


    savepoint prueba;


    select count(*) into cntusu
    from usuario
    where usuci = ciNew and usuid = ciidNew;

    if cntusu = 0 then

        update usuario
        set usuci = ciNew, usuid = ciidNew, usuidant = ciidOld, usuciant =
            ciOld
        where usuci = ciOld and usuid = ciidOld;
    else

        update usuario
        set usuidant = ciidOld, usuciant = ciOld
        where usuci = cinew and usuid = ciidNew;


        delete from usuario
        where usuci = ciold and usuid = ciidold;

    end if;


    update prestacion
    set usu_ci = ciNew, usu_id = ciidNew
    where usu_ci = ciOld and usu_id = ciidOld;


    update actuacion
    set actusuci = cinew, actusuid = ciidnew
    where actusuci = ciold and actusuid = ciidold;
```

```
    update preins
    set pin_usuci = cinew, pin_usuid = ciidnew
    where pin_usuci = ciold and pin_usuid = ciidold;


    update solicitud
    set solusuci = cinew, solusuid = ciidnew
    where solusuci = ciold and solusuid = ciidold;

    delete from hiscliusu
    where hcusuci = ciOld and hcusuid = ciidOld
    and exists (
        select b.hcusuci
        from hiscliusu b
        where b.hcusuci = cinew and b.hcusuid = ciidnew
            and b.hcinstid = a.hcinstid
        );

    update hiscliusu
    set hcusuci = cinew, hcusuid = ciidnew
    where hcusuci = ciold and hcusuid = ciidold;

    commit;
END;


create procedure CHKNROREMITO (inout p_preid bigint(20), inout p_NroRemito
    int(11), inout P_LabId int(11),
    inout p_ExRemito  varchar(500), inout p_preidRemito bigint(20), inout
        p_canRemito smallint(6))
BEGIN
    DECLARE EXIT HANDLER FOR 1328
        begin
            SET p_ExRemito = 'S';
            SET p_preidRemito = 0;
            SET p_canRemito = 2;
        end;
    DECLARE EXIT HANDLER FOR NOT FOUND
        begin
            SET @SWV_Null_Var = 0;
        end;

    INSERT INTO proclog (proc, fechora) VALUES ('CHKNROREMITO', now());

    SET p_ExRemito = 'N';
    SET p_preidRemito = 0;
    SET p_canRemito = 0;

    SELECT Preid , 1 INTO p_preidRemito, p_canRemito
    FROM preins, presentacion
    WHERE CAST( CASE pin_nroremito
                    WHEN 0 THEN null
                    ELSE pin_nroremito
                    END
                AS unsigned) = p_NroRemito
        AND preins.presentaid = presentacion.presentaid AND presentacion.
            LabId = P_LabId AND Preid <> p_preid;
```

```
    SET p_ExRemito = 'S';
END;
//


drop procedure if exists CONSULTA1530//
create procedure CONSULTA1530 (in consulta TEXT, in consulta2 TEXT, in
    consulta3 TEXT, in consulta4 TEXT,
    in consulta5 TEXT, in consulta6 TEXT)
BEGIN
   DECLARE consulta7 VARCHAR(1500);
   DECLARE aux VARCHAR(255);
   DECLARE aux2 VARCHAR(255);
   DECLARE aux3 VARCHAR(255);
   DECLARE aux4 VARCHAR(255);
   DECLARE aux5 VARCHAR(255);
   DECLARE aux6 VARCHAR(255);

   DECLARE EXIT HANDLER FOR NOT FOUND,SQLEXCEPTION
   begin
      rollback;
   end;

   INSERT INTO proclog (proc, fechora) VALUES ('CONSULTA1530', now());

   SET aux  = consulta;
   SET aux2 = consulta2;
   SET aux3 = consulta3;
   SET aux4 = consulta4;
   SET aux5 = consulta5;
   SET aux6 = consulta6;

   CALL p_audit_consulta1530(aux,aux2,aux3,aux4,aux5,aux6);

   if length(consulta) < 254 and length(consulta2) > 0 then
      SET aux = CONCAT(aux,' ');
   end if;

   if length(consulta2) < 254 and length(consulta3) > 0 then
      SET aux2 = CONCAT(aux2,' ');
   end if;

   if length(consulta3) < 254 and length(consulta4) > 0 then
      SET aux3 = CONCAT(aux3,' ');
   end if;

   if length(consulta4) < 254 and length(consulta5) > 0 then
      SET aux4 = CONCAT(aux4,' ');
   end if;

   if length(consulta5) < 254 and length(consulta6) > 0 then
      SET aux5 = CONCAT(aux5,' ');
   end if;

   SET consulta7 = CONCAT(aux,aux2,aux3,aux4,aux5,aux6);

   COMMIT;

   SET @SWV_Stmt =(consulta7);
```

```
   PREPARE SWT_Stmt FROM @SWV_Stmt;
   EXECUTE SWT_Stmt;
   DEALLOCATE PREPARE SWT_Stmt;
   COMMIT;

end;
//

drop procedure if exists P_AUDIT_CONSULTA1530//
create procedure P_AUDIT_CONSULTA1530(in paux  VARCHAR(255),in paux2
    VARCHAR(255), in paux3 VARCHAR(255),
    in paux4 VARCHAR(255), in paux5 VARCHAR(255), in paux6 VARCHAR(255))
begin
    INSERT INTO proclog (proc, fechora) VALUES ('P_AUDIT_CONSULTA1530', now
        ());
    INSERT INTO audit_CONSULTA1530(aud_date, consulta, consulta2, consulta3
        , consulta4, consulta5, consulta6)
    VALUES(CURRENT_TIMESTAMP, paux, paux2, paux3, paux4, paux5, paux6);

   COMMIT;
end;
//


drop procedure if exists CONSULTA3060//
create procedure CONSULTA3060 (in consulta01 TEXT, in consulta02 TEXT, in
    consulta03 TEXT,
    in consulta04 TEXT, in consulta05 TEXT, in consulta06 TEXT, in
        consulta07 TEXT, in consulta08 TEXT,
    in consulta09 TEXT, in consulta10 TEXT, in consulta11 TEXT, in
        consulta12 TEXT)
BEGIN
   DECLARE consulta13 VARCHAR(3060);
   DECLARE aux01 VARCHAR(255);
   DECLARE aux02 VARCHAR(255);
   DECLARE aux03 VARCHAR(255);
   DECLARE aux04 VARCHAR(255);
   DECLARE aux05 VARCHAR(255);
   DECLARE aux06 VARCHAR(255);
   DECLARE aux07 VARCHAR(255);
   DECLARE aux08 VARCHAR(255);
   DECLARE aux09 VARCHAR(255);
   DECLARE aux10 VARCHAR(255);
   DECLARE aux11 VARCHAR(255);
   DECLARE aux12 VARCHAR(255);
   DECLARE EXIT HANDLER FOR NOT FOUND,SQLEXCEPTION
   begin
      rollback;
   end;

   SET aux01 = consulta01;
   SET aux02 = consulta02;
   SET aux03 = consulta03;
   SET aux04 = consulta04;
   SET aux05 = consulta05;
   SET aux06 = consulta06;
   SET aux07 = consulta07;
   SET aux08 = consulta08;
   SET aux09 = consulta09;
```

```
SET aux10 = consulta10;
SET aux11 = consulta11;
SET aux12 = consulta12;

INSERT INTO proclog (proc, fechora) VALUES ('CONSULTA3060', now());
CALL p_audit_consulta3060(aux01,aux02,aux03,aux04,aux05,aux06,aux07,
    aux08,aux09,aux10,aux11,aux12);

if length(consulta01) < 254 and length(consulta02) > 0 then
   SET aux01 = CONCAT(aux01,' ');
end if;
if length(consulta02) < 254 and length(consulta03) > 0 then
   SET aux02 = CONCAT(aux02,' ');
end if;

if length(consulta03) < 254 and length(consulta04) > 0 then
   SET aux03 = CONCAT(aux03,' ');
end if;

if length(consulta04) < 254 and length(consulta05) > 0 then
   SET aux04 = CONCAT(aux04,' ');
end if;

if length(consulta05) < 254 and length(consulta06) > 0 then
   SET aux05 = CONCAT(aux05,' ');
end if;

if length(consulta06) < 254 and length(consulta07) > 0 then
   SET aux06 = CONCAT(aux06,' ');
end if;

if length(consulta07) < 254 and length(consulta08) > 0 then
   SET aux07 = CONCAT(aux07,' ');
end if;
if length(consulta08) < 254 and length(consulta09) > 0 then
   SET aux08 = CONCAT(aux08,' ');
end if;

if length(consulta09) < 254 and length(consulta10) > 0 then
   SET aux09 = CONCAT(aux09,' ');
end if;

if length(consulta10) < 254 and length(consulta11) > 0 then
   SET aux10 = CONCAT(aux10,' ');
end if;

if length(consulta11) < 254 and length(consulta12) > 0 then
   SET aux11 = CONCAT(aux11,' ');
end if;

SET consulta13 = CONCAT(aux01,aux02,aux03,aux04,aux05,aux06,aux07,aux08,
    aux09,aux10,aux11,
aux12);

SET @SWV_Stmt =(consulta13);
PREPARE SWT_Stmt FROM @SWV_Stmt;
EXECUTE SWT_Stmt;
DEALLOCATE PREPARE SWT_Stmt;
COMMIT;
```

```
end;
//

drop procedure if exists P_AUDIT_CONSULTA3060//
create procedure P_AUDIT_CONSULTA3060(in paux  VARCHAR(255), in paux2
    VARCHAR(255), in paux3 VARCHAR(255),
    in paux4 VARCHAR(255),in paux5 VARCHAR(255), in paux6 VARCHAR(255), in
        paux7 VARCHAR(255), in paux8 VARCHAR(255),
    in paux9 VARCHAR(255),in paux10 VARCHAR(255),in paux11 VARCHAR(255), in
         paux12 VARCHAR(255))
begin

    INSERT INTO proclog (proc, fechora) VALUES ('P_AUDIT_CONSULTA3060', now
        ());

    INSERT INTO audit_CONSULTA3060(aud_date, consulta, consulta2, consulta3
        , consulta4, consulta5, consulta6, consulta7, consulta8, consulta9,
         consulta10, consulta11, consulta12)
    VALUES(CURRENT_TIMESTAMP, paux, paux2, paux3, paux4, paux5, paux6,
        paux7, paux8, paux9, paux10, paux11, paux12);

   COMMIT;
end;
//

drop procedure if exists CONSULTAACTUACION//
CREATE PROCEDURE CONSULTAACTUACION (in proveeduria char, in fchIni char, in
     fchFin char, in pcenid int(11),
    in puniid int(11), in pfamid varchar(8),in pfamtpo varchar(3), in
        psubfamid varchar(8), in psusid varchar(8),
    in pinsid smallint(6), in ppresentaid bigint(20), inout totRec varchar
        (8), inout totExp varchar(8))
BEGIN
   DECLARE consulta VARCHAR(1000);

   DECLARE ccenid int(11);
   DECLARE cuniid int(11);
   DECLARE cfchIni TEXT;
   DECLARE cfchFin TEXT;
   DECLARE cfamid VARCHAR(8);
   DECLARE cfamtpo VARCHAR(3);
   DECLARE csubfamid VARCHAR(8);
   DECLARE csusid VARCHAR(8);
   DECLARE cinsid SMALLINT(6);
   DECLARE cpresentaid BIGINT(20);

   DECLARE NO_DATA INT DEFAULT 0;

   DECLARE v_estid VARCHAR(255);
   DECLARE v_traid VARCHAR(255);
   DECLARE v_actsaldomovimiento VARCHAR(255);
   DECLARE v_actsaldoanterior VARCHAR(255);
   DECLARE v_actsaldoposterior VARCHAR(255);
   DECLARE v_actfamid VARCHAR(255);
   DECLARE v_actfamtpo VARCHAR(255);
   DECLARE v_actsubfamid VARCHAR(255);
   DECLARE v_actsusid VARCHAR(255);
   DECLARE v_actinsid VARCHAR(255);
```

```
DECLARE v_preid VARCHAR(255);

DECLARE C_ACTUACION  CURSOR FOR SELECT
     estid,traid,actsaldomovimiento,actsaldoanterior,actsaldoposterior,
         actfamid, actfamtpo,
     actsubfamid, actsusid, actinsid, preid
FROM actuacion
WHERE objwid = 6 and actfchini >= STR_TO_DATE(cfchIni,'%d/%m/%Y') and
    actfchini <  STR_TO_DATE(cfchFin,'%d/%m/%Y') and
 actcenid = ccenid and actuniid = cuniid and actfamid = cfamid and
     actfamtpo = cfamtpo and actsubfamid = csubfamid and
 actsusid = csusid and actinsid = cinsid and actpresentaid = cpresentaid
      and estid >= 2215 and estid <= 2221;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET NO_DATA = -1;

SET totRec = 0;
SET totExp = 0;

SET ccenid = pcenid;
SET cuniid = puniid;
SET cfchIni = fchini;
SET cfchFin = fchfin;
SET cfamid = pfamid;
SET cfamtpo = pfamtpo;
SET csubfamid = psubfamid;
SET csusid = psusid;
SET cinsid = pinsid;
SET cpresentaid = ppresentaid;

INSERT INTO proclog (proc, fechora) VALUES ('CONSULTAACTUACION', now());

OPEN C_ACTUACION;

SET NO_DATA = 0;

FETCH C_ACTUACION INTO v_estid,v_traid,v_actsaldomovimiento,
    v_actsaldoanterior,v_actsaldoposterior, v_actfamid,v_actfamtpo,
     v_actsubfamid,v_actsusid,v_actinsid,v_preid;

WHILE NO_DATA = 0 DO

   IF (v_estid = 2217 and proveeduria = 'N' and (v_traid = 1003 or
       v_traid = 1005 or v_traid = 1018 or v_traid = 1019 or v_traid =
       2040 or v_traid = 1050 or v_traid = 1049))
   or (v_estid = 2218 and proveeduria = 'S' and (v_traid = 1003 or
       v_traid = 1005 or v_traid = 1031 or v_traid = 1032  or v_traid =
       1050 or v_traid = 1049)) then
      IF v_actsaldoposterior > v_actsaldoanterior THEN
         SET totRec = totRec -v_actsaldomovimiento;
      ELSE
         SET totRec = CONCAT(totRec,v_actsaldomovimiento);
      END IF;
   ELSE
      IF (v_estid = 2215 and proveeduria = 'N') or (v_estid = 2221 and
          proveeduria = 'S') THEN
         SET totExp = CONCAT(totExp,v_actsaldomovimiento);
      END IF;
   END IF;
```

```
      FETCH C_ACTUACION INTO v_estid,v_traid,v_actsaldomovimiento,
          v_actsaldoanterior,v_actsaldoposterior, v_actfamid,v_actfamtpo,
        v_actsubfamid, v_actsusid,v_actinsid,v_preid;
   END WHILE;

   CLOSE C_ACTUACION;
END;
//

drop procedure if exists CONSULTAMOVIMIENTO;
create procedure CONSULTAMOVIMIENTO (in fchIni char, in fchFin char, in
    pcenid int(11),in puniid int(11), in pfamid varchar(8),in pfamtpo
    varchar(3),
    in psubfamid varchar(8), in psusid varchar(8), in pinsid smallint(6),
        in ppresentaid bigint(20), in sesId char, inout totRec varchar(8),
        inout totExp varchar(8))
BEGIN

    DECLARE consulta VARCHAR(1000);
    DECLARE ccenid int(11);
    DECLARE cuniid int(11);
    DECLARE cfchIni TEXT;
    DECLARE cfchFin TEXT;
    DECLARE cfamid VARCHAR(8);
    DECLARE cfamtpo VARCHAR(3);
    DECLARE csubfamid VARCHAR(8);
    DECLARE csusid VARCHAR(8);
    DECLARE cinsid SMALLINT(6);
    DECLARE cpresentaid BIGINT(20);
    DECLARE NO_DATA INT DEFAULT 0;
    DECLARE v_estid VARCHAR(255);
    DECLARE v_traid VARCHAR(255);
    DECLARE v_actsaldomovimiento VARCHAR(255);

    DECLARE C_ACTUACION  CURSOR FOR SELECT estid,traid,actsaldomovimiento
    FROM mirroractuacion
    WHERE sesion = sesId and objwid = 6 and actfchini >= STR_TO_DATE(
        cfchIni,'%d/%m/%Y') and
        actfchini <  STR_TO_DATE(cfchFin,'%d/%m/%Y') and actcenidpuesto =
            ccenid and
        actuniidpuesto = cuniid and actfamid = cfamid and actfamtpo =
            cfamtpo and
        actsubfamid = csubfamid and actsusid = csusid and actinsid = cinsid
             and
        actpresentaid = cpresentaid and estid >= 2003 and estid <= 2229;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET NO_DATA = -1;

    SET totRec = 0;
    SET totExp = 0;

    SET ccenid = pcenid;
    SET cuniid = puniid;
    SET cfchIni = fchini;
    SET cfchFin = fchfin;
    SET cfamid = pfamid;
    SET cfamtpo = pfamtpo;
    SET csubfamid = psubfamid;
```

```
    SET csusid = psusid;
    SET cinsid = pinsid;
    SET cpresentaid = ppresentaid;

    SET NO_DATA = 0;

    INSERT INTO proclog (proc, fechora) VALUES ('CONSULTAMOVIMIENTO', now()
        );


    OPEN C_ACTUACION;
    FETCH C_ACTUACION INTO v_estid, v_traid, v_actsaldomovimiento;

    WHILE NO_DATA = 0 DO
        IF v_estid = 2003 THEN
            SET totRec = CONCAT(totRec,v_actsaldomovimiento);
        ELSE
            IF v_estid = 2213 or v_estid = 2216 THEN
                IF v_estid = 2216 and (v_traid = 1032 or v_traid = 1035)
                    THEN
                    SET totExp = totExp - v_actsaldomovimiento;
                ELSE
                    SET totExp = CONCAT(totExp,v_actsaldomovimiento);
                END IF;
            ELSE
                IF v_estid = 2004 THEN
                    SET totRec = totRec - v_actsaldomovimiento;
                ELSE
                    IF v_estid = 2214 or v_estid = 2229 THEN
                        SET totRec = CONCAT(totRec,v_actsaldomovimiento);
                    ELSE
                        IF v_estid = 2228 and (v_traid = 2090 or v_traid =
                            2092) THEN
                            SET totExp = CONCAT(totExp,v_actsaldomovimiento
                                );
                        END IF;
                    END IF;
                END IF;
            END IF;
        END IF;

        FETCH C_ACTUACION INTO v_estid, v_traid, v_actsaldomovimiento;
    END WHILE;
    CLOSE C_ACTUACION;
END;
//

CREATE PROCEDURE CONSULTAMOVIMIENTO_FDES (in fchIni char, in fchFin char,
    in pcenid int(11),in puniid int(11),
    in pfamid varchar(8),in pfamtpo varchar(3), in psubfamid varchar(8), in
         psusid varchar(8), in pinsid smallint(6),
    in ppresentaid bigint(20), in sesId char, inout totRec varchar(8),
        inout totExp varchar(8))
BEGIN
    DECLARE consulta VARCHAR(1000);
    DECLARE ccenid INT(11);
    DECLARE cuniid INT(11);
    DECLARE cfchIni TEXT;
    DECLARE cfchFin TEXT;
```

```
DECLARE cfamid VARCHAR(8);
DECLARE cfamtpo VARCHAR(3);
DECLARE csubfamid VARCHAR(8);
DECLARE csusid VARCHAR(8);
DECLARE cinsid SMALLINT(6);
DECLARE cpresentaid BIGINT(20);
DECLARE NO_DATA INT DEFAULT 0;
DECLARE v_estid VARCHAR(255);
DECLARE v_traid VARCHAR(255);
DECLARE v_actsaldomovimiento VARCHAR(255);

DECLARE C_ACTUACION CURSOR FOR SELECT estid,traid,actsaldomovimiento
FROM mirroractuacion
WHERE sesion = sesId and objwid = 6 anD actfchini >= STR_TO_DATE(
    cfchIni,'%d/%m/%Y') and actfchini <  STR_TO_DATE(cfchFin,'%d/%m/%Y'
    ) and
    actcenid = ccenid and actuniid = cuniid and actfamid = cfamid and
        actfamtpo = cfamtpo and actsubfamid = csubfamid and
    actsusid = csusid and actinsid = cinsid and actpresentaid =
        cpresentaid and estid in(2216,2228);

DECLARE CONTINUE HANDLER FOR NOT FOUND SET NO_DATA = -1;

SET totRec = 0;
SET totExp = 0;
SET ccenid = pcenid;
SET cuniid = puniid;
SET cfchIni = fchini;
SET cfchFin = fchfin;
SET cfamid = pfamid;
SET cfamtpo = pfamtpo;
SET csubfamid = psubfamid;
SET csusid = psusid;
SET cinsid = pinsid;
SET cpresentaid = ppresentaid;

INSERT INTO proclog (proc, fechora) VALUES ('CONSULTAMOVIMIENTO_FDES',
    now());

OPEN C_ACTUACION;
SET NO_DATA = 0;
FETCH C_ACTUACION INTO v_estid,v_traid,v_actsaldomovimiento;

WHILE NO_DATA = 0 DO
    if v_estid = 2216 then
        if v_traid <> 1019 and v_traid <> 1035 then
            SET totExp = CONCAT(totExp, v_actsaldomovimiento);
        else
            SET totExp = totExp - v_actsaldomovimiento;
        end if;
    else
        if v_traid = 2093 then
            SET totExp = CONCAT(totExp, v_actsaldomovimiento);
        end if;
    end if;
    SET NO_DATA = 0;
    FETCH C_ACTUACION INTO v_estid, v_traid, v_actsaldomovimiento;
END WHILE;
```

```
   SET NO_DATA = 0;
   CLOSE C_ACTUACION;
END;
//


CREATE PROCEDURE CONSULTAMOVIMIENTO_FEX (in fchIni char, in fchFin char, in
     pcenid int(11),in puniid int(11),in pfamid varchar(8),
    in pfamtpo varchar(3), in psubfamid varchar(8), in psusid varchar(8),
        in pinsid smallint(6), in ppresentaid bigint(20),
    in sesId char, inout totRec varchar(8), inout totExp varchar(8))
BEGIN
    DECLARE consulta VARCHAR(1000);
    DECLARE ccenid INT(11);
    DECLARE cuniid INT(11);
    DECLARE cfchIni TEXT;
    DECLARE cfchFin TEXT;
    DECLARE cfamid VARCHAR(8);
    DECLARE cfamtpo VARCHAR(3);
    DECLARE csubfamid VARCHAR(8);
    DECLARE csusid VARCHAR(8);
    DECLARE cinsid SMALLINT(6);
    DECLARE cpresentaid BIGINT(20);
    DECLARE NO_DATA INT DEFAULT 0;
    DECLARE v_estid VARCHAR(255);
    DECLARE v_traid VARCHAR(255);
    DECLARE v_actsaldomovimiento VARCHAR(255);

    DECLARE C_ACTUACION CURSOR FOR SELECT estid,traid,actsaldomovimiento
    FROM mirroractuacion
    WHERE sesion = sesId and objwid = 6 and actfchini >= STR_TO_DATE(
        cfchIni,'%d/%m/%Y') and actfchini <  STR_TO_DATE(cfchFin,'%d/%m/%Y'
        ) and
        actcenid = ccenid and actuniid = cuniid and actfamid = cfamid and
            actfamtpo = cfamtpo and actsubfamid = csubfamid and
        actsusid = csusid and actinsid = cinsid and actpresentaid =
            cpresentaid and estid in(2213,2214);

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET NO_DATA = -1;

    SET totRec = 0;
    SET totExp = 0;
    SET ccenid = pcenid;
    SET cuniid = puniid;
    SET cfchIni = fchini;
    SET cfchFin = fchfin;
    SET cfamid = pfamid;
    SET cfamtpo = pfamtpo;
    SET csubfamid = psubfamid;
    SET csusid = psusid;
    SET cinsid = pinsid;
    SET cpresentaid = ppresentaid;

    INSERT INTO proclog (proc, fechora) VALUES ('CONSULTAMOVIMIENTO_FEX',
        now());

    OPEN C_ACTUACION;
    SET NO_DATA = 0;
    FETCH C_ACTUACION INTO v_estid, v_traid, v_actsaldomovimiento;
```

```
    WHILE NO_DATA = 0 DO
        if  v_estid = 2213 and v_traid <> 2040 then
            SET totRec = CONCAT(totRec, v_actsaldomovimiento);
        else
            if v_estid = 2214 and v_traid <> 2098 then
                SET totRec = totRec - v_actsaldomovimiento;
            end if;
        end if;
        SET NO_DATA = 0;
        FETCH C_ACTUACION INTO v_estid, v_traid, v_actsaldomovimiento;
    END WHILE;

    SET NO_DATA = 0;
    CLOSE C_ACTUACION;
END;
//

CREATE PROCEDURE CONSULTOAJUSTE (in fchIni char, in fchFin char, in pcenid
    int(11),in puniid int(11), in pfamid varchar(8),in pfamtpo varchar(3),
    in psubfamid varchar(8), in psusid varchar(8), in pinsid smallint(6),
        in ppresentaid bigint(20), in sesId char, inout totAju varchar(8),
        inout totdes varchar(8))
BEGIN

    DECLARE consulta VARCHAR(1000);
    DECLARE ccenid INT(11);
    DECLARE cuniid INT(11);
    DECLARE cfchIni TEXT;
    DECLARE cfchFin TEXT;
    DECLARE cfamid VARCHAR(8);
    DECLARE cfamtpo VARCHAR(3);
    DECLARE csubfamid VARCHAR(8);
    DECLARE csusid VARCHAR(8);
    DECLARE cinsid SMALLINT(6);
    DECLARE cpresentaid BIGINT(20);
    DECLARE NO_DATA INT DEFAULT 0;
    DECLARE v_estid VARCHAR(255);
    DECLARE v_traid VARCHAR(255);
    DECLARE v_actsaldomovimiento VARCHAR(255);

    DECLARE C_ACTUACION  CURSOR FOR SELECT estid,traid,actsaldomovimiento
    FROM mirroractuacion
    WHERE sesion = sesId and objwid = 6 and actfchini >= STR_TO_DATE(
        cfchIni,'%d/%m/%Y') and actfchini <  STR_TO_DATE(cfchFin,'%d/%m/%Y'
        ) and
        actcenidpuesto = ccenid and actuniidpuesto = cuniid and actfamid =
            cfamid and actfamtpo = cfamtpo and actsubfamid = csubfamid and
        actsusid = csusid and actinsid = cinsid and actpresentaid =
            cpresentaid and estid in(2002,2224);

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET NO_DATA = -1;

    SET totAju = 0;
    SET totdes = 0;

    SET ccenid = pcenid;
    SET cuniid = puniid;
    SET cfchIni = fchini;
```

```
    SET cfchFin = fchfin;
    SET cfamid = pfamid;
    SET cfamtpo = pfamtpo;
    SET csubfamid = psubfamid;
    SET csusid = psusid;
    SET cinsid = pinsid;
    SET cpresentaid = ppresentaid;

    INSERT INTO proclog (proc, fechora) VALUES ('CONSULTOAJUSTE', now());

    OPEN C_ACTUACION;
    SET NO_DATA = 0;
    FETCH C_ACTUACION INTO v_estid, v_traid, v_actsaldomovimiento;

    WHILE NO_DATA = 0 DO
        if v_estid = 2002 and v_traid = 2035 then
            SET totAju = CONCAT(totAju,v_actsaldomovimiento);
        else
            if v_estid = 2002 and v_traid = 2055 then
                SET totAju = totAju -v_actsaldomovimiento;
            else
                if v_estid = 2224 and v_traid = 2059 then
                    SET totdes = CONCAT(totdes,v_actsaldomovimiento);
                else
                    if v_estid = 2224 and v_traid = 2067 then
                        SET totdes = totdes -v_actsaldomovimiento;
                    end if;
                end if;
            end if;
        end if;
        SET NO_DATA = 0;
        FETCH C_ACTUACION INTO v_estid,v_traid,v_actsaldomovimiento;
    END WHILE;

    SET NO_DATA = 0;
    CLOSE C_ACTUACION;
END;
//


drop procedure if exists CONSULTOAJUSTE_FEX//
create procedure CONSULTOAJUSTE_FEX(in fchini TEXT, in fchfin TEXT, in
    pcenid VARCHAR(255),in puniid VARCHAR(255),
    in pfamid VARCHAR(255),in pfamtpo VARCHAR(255), in psubfamid VARCHAR
        (255), in psusid VARCHAR(255), in pinsid VARCHAR(255),
    in ppresentaid VARCHAR(255), in sesId TEXT, INOUT totAju VARCHAR(255),
        INOUT totdes VARCHAR(255))
BEGIN

    DECLARE consulta VARCHAR(1000);

    DECLARE ccenid INT(11);
    DECLARE cuniid INT(11);
    DECLARE cfchIni TEXT;
    DECLARE cfchFin TEXT;
    DECLARE cfamid VARCHAR(8);
    DECLARE cfamtpo VARCHAR(3);
    DECLARE csubfamid VARCHAR(8);
    DECLARE csusid VARCHAR(8);
```

```
DECLARE cinsid SMALLINT(6);
DECLARE cpresentaid BIGINT(20);
DECLARE NO_DATA INT DEFAULT 0;
DECLARE v_estid VARCHAR(255);
DECLARE v_traid VARCHAR(255);
DECLARE v_actsaldomovimiento VARCHAR(255);

DECLARE C_ACTUACION  CURSOR FOR SELECT estid,traid,actsaldomovimiento
FROM mirroractuacion
WHERE sesion = sesId and objwid = 6 and actfchini >= STR_TO_DATE(
    cfchIni,'%d/%m/%Y') and actfchini <  STR_TO_DATE(cfchFin,'%d/%m/%Y'
    ) and
    actcenidpuesto = ccenid and actuniidpuesto = cuniid and actfamid =
        cfamid and actfamtpo = cfamtpo and actsubfamid = csubfamid and
    actsusid = csusid and actinsid = cinsid and actpresentaid =
        cpresentaid and estid in(2002,2225);

DECLARE CONTINUE HANDLER FOR NOT FOUND SET NO_DATA = -1;

SET totAju = 0;
SET totdes = 0;

SET ccenid = pcenid;
SET cuniid = puniid;
SET cfchIni = fchini;
SET cfchFin = fchfin;
SET cfamid = pfamid;
SET cfamtpo = pfamtpo;
SET csubfamid = psubfamid;
SET csusid = psusid;
SET cinsid = pinsid;
SET cpresentaid = ppresentaid;

SET NO_DATA = 0;

INSERT INTO proclog (proc, fechora) VALUES ('CONSULTOAJUSTE_FEX', now()
    );

OPEN C_ACTUACION;

FETCH C_ACTUACION INTO v_estid, v_traid, v_actsaldomovimiento;
WHILE NO_DATA = 0 DO
    IF v_estid = 2002 and v_traid = 2034 THEN
        SET totAju = CONCAT(totAju,v_actsaldomovimiento);
    ELSE
        IF v_estid = 2002 and v_traid = 2058 THEN
            SET totAju = totAju - v_actsaldomovimiento;
        ELSE
            IF v_estid = 2225 and v_traid = 2060 THEN
                SET totdes = CONCAT(totdes, v_actsaldomovimiento);
            ELSE
                IF v_estid = 2225 and v_traid = 2075 THEN
                    SET totdes = totdes - v_actsaldomovimiento;
                END IF;
            END IF;
        END IF;
    END IF;
    FETCH C_ACTUACION INTO v_estid, v_traid, v_actsaldomovimiento;
END WHILE;
```

```
   CLOSE C_ACTUACION;
END;
//


CREATE PROCEDURE COPIOSALESTADO()
BEGIN

   DECLARE EXIT HANDLER FOR NOT FOUND,SQLEXCEPTION
   begin
      rollback;
   end;

   SAVEPOINT prueba;

   INSERT INTO hsalestado
   SELECT CURRENT_TIMESTAMP,a.cenid,a.uniid,traobjwestid,presentaid,
       salfamid,salfamtpo,salsusid,salinsid,sallote,salsubfamid,
       salestactual,salestactcomp,salestfch
   FROM salestado a, centro b
   WHERE traobjwestid IN(2217,2218) AND b.cenid = a.cenid AND instid IN
       (1,3);

   COMMIT;

end;
//


CREATE PROCEDURE CREOAUXILIAR(in sesId TEXT,in fchini TEXT, in fchfin TEXT,
     INOUT pcenid VARCHAR(255),INOUT puniid VARCHAR(255))
BEGIN

    DECLARE cantidad INT;

    DELETE FROM mirroractuacion
    WHERE sesion = sesId;

    COMMIT;

    INSERT INTO proclog (proc, fechora) VALUES ('CONSULTOAJUSTE_FEX', now()
        );

    INSERT INTO mirroractuacion
    SELECT  ACTID,OBJWID,PREID,ACTFCHINI,ACTFCHFIN, ACTOBS,TRAID,ACTUSR,
        ESTID,ACTPREDIAG,ACTPREDIAGDSC,ACTMOVID,ACTCENID,ACTUNIID,
        ACTCLAVEPROPUESTA,ACTCLAVEINFORMADA,ACTCLAVECONFIRMADA,ACTUSUCI,
            ACTUSUID,ACTEMEPREDIAG, ACTEMEPREDIAGDSC,ACTEMECENIDTRAS,
        ACTPINPRECIO,ACTSALDOMOVIMIENTO,ACTSALDOPOSTERIOR, ACTSALDOANTERIOR
            ,ACTPRESENTAID,ACTINSID,ACTSUSID,ACTSUBFAMID,ACTFAMTPO,
        ACTFAMID,ACTOPERACION, ACTUNIIDPUESTO,ACTCENIDPUESTO,ACTPREMOTID,
            ACTPREMOTDSC,sesId,CURRENT_TIMESTAMP
    FROM actuacion
    WHERE objwid = 6 and actfchini >= STR_TO_DATE(fchini,'%d/%m/%Y') and
        actfchini <  STR_TO_DATE(fchfin,'%d/%m/%Y') and
        actcenidpuesto = pcenid and actuniidpuesto = puniid and estid in
            (2003,2213,2004,2214,2228,2002,2224,2216,2229);
```

```
   COMMIT;
END;
//


CREATE PROCEDURE CREOAUXILIAR_AA(in sesId TEXT,in fchini TEXT, in fchfin
    TEXT, INOUT pcenid VARCHAR(255),INOUT puniid VARCHAR(255),  INOUT
    v_err_num INT)
BEGIN
    DECLARE cantidad INT;
    DECLARE v_session VARCHAR(40);

    SET v_session = SUBSTRING(Rtrim(sesId),1,40);
    SET v_err_num = 0;

    delete from AuxAct
    where AASesid = v_session;

    COMMIT;

    BEGIN

        DECLARE EXIT HANDLER FOR NOT FOUND,SQLEXCEPTION
        begin
            SET v_err_num = -1;
        end;
        insert into AuxAct(AASESID,AAACTID,AACENID,AAUNIID,AAACTCENIDPUESTO
            ,AAACTUNIIDPUESTO,AAINI,AAFIN, AAUSUCI,AAUSUID,
            AAPRESENTAID,AAFAMID,AAFAMTPO,AASUBFAMID,AASUSID,AAINSID,
                AASALDOMOV, AASALDOPOST, AASALDOANT, AAACTPINPRECIO,
            AAPREID, AATRAID,AAESTID,AAUSR)
        select  v_session , ACTID,ACTCENID,ACTUNIID, ACTCENIDPUESTO,
            ACTUNIIDPUESTO, ACTFCHINI,ACTFCHFIN, ActUsuCI ,  ActUsuId,
            ACTPRESENTAID, ACTFAMID,ACTFAMTPO,ACTSUBFAMID,ACTSUSID,ACTINSID
                , ACTSALDOMOVIMIENTO,ACTSALDOPOSTERIOR,ACTSALDOANTERIOR,
            ACTPINPRECIO,PREID, TRAID,ESTID,ACTUSR
        FROM actuacion
        WHERE objwid = 6 and actfchini >= STR_TO_DATE(fchini,'%d/%m/%Y')
            and actfchini <  STR_TO_DATE(fchfin,'%d/%m/%Y') and
            actcenidpuesto = pcenid and actuniidpuesto = puniid and estid
                in(2003,2213,2004,2214,2228,2002,2224,2216,2229) and
      ACTFAMTPO = 'MED';
    END;

   COMMIT;
END;
//

CREATE PROCEDURE CREOAUXILIAR_FEX(in sesId TEXT,in fchini TEXT, in fchfin
    TEXT, INOUT pcenid VARCHAR(255),INOUT puniid VARCHAR(255),
    INOUT vadid VARCHAR(255))

BEGIN
   DECLARE cantidad INT;

   delete from mirroractuacion
   where sesion = sesId;

   COMMIT;
```

```
    INSERT INTO proclog (proc, fechora) VALUES ('CREOAUXILIAR_FEX', now());

   if vadid <> 99 then
        INSERT INTO mirroractuacion
        SELECT ACTID,OBJWID,PREID,ACTFCHINI,ACTFCHFIN, ACTOBS,TRAID,ACTUSR,
            ESTID,ACTPREDIAG,ACTPREDIAGDSC,ACTMOVID,ACTCENID,ACTUNIID,
            ACTCLAVEPROPUESTA,ACTCLAVEINFORMADA,ACTCLAVECONFIRMADA,ACTUSUCI
                ,ACTUSUID,ACTEMEPREDIAG,ACTEMEPREDIAGDSC,ACTEMECENIDTRAS,
            ACTPINPRECIO,ACTSALDOMOVIMIENTO,ACTSALDOPOSTERIOR,
                ACTSALDOANTERIOR,ACTPRESENTAID,ACTINSID,ACTSUSID,
                ACTSUBFAMID,ACTFAMTPO,
            ACTFAMID,ACTOPERACION,ACTUNIIDPUESTO,ACTCENIDPUESTO,ACTPREMOTID
                ,ACTPREMOTDSC,sesId,CURRENT_TIMESTAMP
        FROM actuacion
        WHERE objwid = 6 and actfchini >= STR_TO_DATE(fchini,'%d/%m/%Y')
            and actfchini <  STR_TO_DATE(fchfin,'%d/%m/%Y') and
            actcenid = pcenid and actuniid = puniid and estid in
                (2213,2214,2216,2002,2225,2228) and
            (actfamid,actfamtpo,actsubfamid,actsusid,actinsid) in(select
                famid,famtpo,subfamid,susid,insid from vademecumins
             where vadid = vadid);
    else
        insert into mirroractuacion
        SELECT ACTID,OBJWID,PREID,ACTFCHINI,ACTFCHFIN, ACTOBS,TRAID,ACTUSR,
            ESTID,ACTPREDIAG,ACTPREDIAGDSC,ACTMOVID,ACTCENID,ACTUNIID,
            ACTCLAVEPROPUESTA,ACTCLAVEINFORMADA,ACTCLAVECONFIRMADA,ACTUSUCI
                ,ACTUSUID,ACTEMEPREDIAG,ACTEMEPREDIAGDSC,ACTEMECENIDTRAS,
            ACTPINPRECIO,ACTSALDOMOVIMIENTO,ACTSALDOPOSTERIOR,
                ACTSALDOANTERIOR,ACTPRESENTAID,ACTINSID,ACTSUSID,
                ACTSUBFAMID,ACTFAMTPO,
            ACTFAMID,ACTOPERACION, ACTUNIIDPUESTO,ACTCENIDPUESTO,
                ACTPREMOTID,ACTPREMOTDSC,sesId,CURRENT_TIMESTAMP
      FROM actuacion
      WHERE objwid = 6 and actfchini >= STR_TO_DATE(fchini,'%d/%m/%Y') and
          actfchini <  STR_TO_DATE(fchfin,'%d/%m/%Y') and
      actcenid = pcenid and actuniid = puniid and estid in
          (2213,2214,2216,2002,2225) and
      (actfamid,actfamtpo,actsubfamid,actsusid,actinsid) in(select famid,
          famtpo,subfamid,susid,insid from vademecumins
       where vadid in(1,3));
    end if;
   COMMIT;
END;
//

CREATE PROCEDURE DISTRIBUCION()
BEGIN

1   DECLARE i INT;
   DECLARE v_fec DATETIME;

   SET @SWV_Stmt = 'TRUNCATE TABLE disedad';
   PREPARE SWT_Stmt FROM @SWV_Stmt;
   EXECUTE SWT_Stmt;
   DEALLOCATE PREPARE SWT_Stmt;

   COMMIT;
```

```
SET v_fec = STR_TO_DATE('31/12/2007','%d/%m/%Y');


INSERT INTO disedad
SELECT '<1',ususexo,COUNT(*)
FROM borrar_100107_b, usuario



WHERE usuci = usu_ci and usuid = usu_id and not usufchnac is null and ((
    datediff(v_fec,usufchnac)/365) < 1)
group by ususexo;


INSERT INTO disedad
SELECT '1a4',ususexo,COUNT(*)
FROM borrar_100107_b, usuario



WHERE usuci = usu_ci AND usuid = usu_id AND NOT usufchnac IS NULL and ((
    datediff(v_fec,usufchnac)/365) >= 1) and ((datediff(v_fec,usufchnac)
    /365) < 5)
group by ususexo;

SET i = 5;
WHILE i <= 80 DO
   IF i < 10 THEN

     INSERT INTO disedad
     SELECT CONCAT('0',i,'a',i+5 -1),ususexo,COUNT(*)
     FROM borrar_100107_b, usuario



     WHERE usuci = usu_ci and usuid = usu_id and not usufchnac is null
         and ((datediff(v_fec,usufchnac)/365) >= i) and ((datediff(v_fec
         ,usufchnac)/365) < i+5)
     group by ususexo;
   else

     insert into disedad
     select CONCAT(i,'a',i+5 -1),ususexo,count(*)
     from borrar_100107_b, usuario



     where usuci = usu_ci and usuid = usu_id and not usufchnac is null
         and ((datediff(v_fec,usufchnac)/365) >= i) and ((datediff(v_fec
         ,usufchnac)/365) < i+5)
     group by ususexo;
   end if;

   COMMIT;

   SET i = i+5;
END WHILE;
```

```
   insert into disedad
   select '85+',ususexo,count(*)
   from borrar_100107_b, usuario


   where usuci = usu_ci and usuid = usu_id and not usufchnac is null and ((
       datediff(v_fec,usufchnac)/365) >= 85)
   group by ususexo;
end;
//

CREATE PROCEDURE EJECUTOCONSULTA(in consulta TEXT,in consulta2 TEXT,in
    consulta3 TEXT,in consulta4 TEXT, in consulta5 TEXT)
begin
   DECLARE consulta6 VARCHAR(1500);
   DECLARE aux VARCHAR(255);
   DECLARE aux2 VARCHAR(255);
   DECLARE aux3 VARCHAR(255);
   DECLARE aux4 VARCHAR(255);
   DECLARE aux5 VARCHAR(255);
   DECLARE EXIT HANDLER FOR NOT FOUND,SQLEXCEPTION
   begin
      rollback;
   end;

   SET aux  = consulta;
   SET aux2 = consulta2;
   SET aux3 = consulta3;
   SET aux4 = consulta4;
   SET aux5 = consulta5;

   INSERT INTO proclog (proc, fechora) VALUES ('EJECUTOCONSULTA', now());

   CALL p_audit_ejecutoconsulta(aux,aux2,aux3,aux4,aux5);

   if length(consulta) < 254 and length(consulta2) > 0 then
      SET aux = CONCAT(aux,' ');
   end if;

   if length(consulta2) < 254 and length(consulta3) > 0 then
      SET aux2 = CONCAT(aux2,' ');
   end if;

   if length(consulta3) < 254 and length(consulta4) > 0 then
      SET aux3 = CONCAT(aux3,' ');
   end if;

   if length(consulta4) < 254 and length(consulta5) > 0 then
      SET aux4 = CONCAT(aux4,' ');
   end if;

   SET consulta6 = CONCAT(aux,aux2,aux3,aux4,aux5);


   SET @SWV_Stmt =(consulta6);
   PREPARE SWT_Stmt FROM @SWV_Stmt;
```

```
   EXECUTE SWT_Stmt;
   DEALLOCATE PREPARE SWT_Stmt;
   COMMIT;

end;
//

drop procedure if exists ELIMINOUSUARIO;
CREATE PROCEDURE ELIMINOUSUARIO (in ciborrar VARCHAR(255),in idborrar
    VARCHAR(255), out error TEXT)
begin
   DECLARE cntusu MEDIUMINT;

   DECLARE EXIT HANDLER FOR NOT FOUND,SQLEXCEPTION
   begin
      ROLLBACK TO SAVEPOINT prueba;
   end;

   SET error = 'N';

   INSERT INTO proclog (proc, fechora) VALUES ('ELIMINOUSUARIO', now());

   SAVEPOINT prueba;


   SELECT count(*) INTO cntusu
   FROM prestacion
   WHERE usu_ci = ciborrar AND usu_id = idborrar AND preserid <> 10002;

   if cntusu = 0 then


      DELETE FROM usuario
      WHERE usuci = ciborrar AND usuid = idborrar;


      DELETE FROM prestacion
      WHERE usu_ci = ciborrar AND usu_id = idborrar;


      DELETE FROM actuacion
      WHERE actusuci = ciborrar AND actusuid = idborrar;


      DELETE FROM preins
      WHERE pin_usuci = ciborrar AND pin_usuid = idborrar;


      DELETE FROM hiscliusu
      WHERE hcusuci = ciborrar AND hcusuid = idborrar;

      COMMIT;
   else
      SET error = 'S';
   end if;

   COMMIT;

end;
```

```
//


CREATE PROCEDURE EXECUTE_IMMEDIATE(p_sql_text VARCHAR(4000))
BEGIN

   DECLARE l_cursor INT DEFAULT 0;
   DECLARE rc INT DEFAULT 0;
   DECLARE stmt VARCHAR(1000);

   DECLARE COMPILATION_ERROR CONDITION FOR SQLSTATE '24344';


   DECLARE EXIT HANDLER FOR COMPILATION_ERROR
   begin
      CALL CLOSE_CURSOR(l_cursor);
   end;

   DECLARE EXIT HANDLER FOR NOT FOUND,SQLEXCEPTION
   begin
      DECLARE SWV_MESSAGE_TEXT VARCHAR(4000);
      CALL CLOSE_CURSOR(l_cursor);
      SET SWV_MESSAGE_TEXT = CONCAT('Internal error','SQLWAYS_EVAL# ''',
          p_sql_text,'''   ');
      SIGNAL SQLSTATE '20101'
      SET MESSAGE_TEXT = SWV_MESSAGE_TEXT;
   end;

   SET l_cursor = DBMS_SQL.OPEN_CURSOR;
   CALL PARSE(l_cursor,p_sql_text,DBMS_SQL.NATIVE);
   SET rc = `EXECUTE`(l_cursor);
   CALL CLOSE_CURSOR(l_cursor);

END;
//

drop procedure if exists EXISTE_HORA;
CREATE PROCEDURE EXISTE_HORA(in p_cenid VARCHAR(255), in p_uniid VARCHAR
    (255), in p_funid VARCHAR(255), in p_serid VARCHAR(255),
    in p_desde VARCHAR(4000), in p_hasta VARCHAR(4000), out p_existe INT)
begin
   DECLARE v_PreHorFchD_ini VARCHAR(255);
   DECLARE v_PreHorFchD_fin VARCHAR(255);

   DECLARE EXIT HANDLER FOR NOT FOUND
   begin
      SET p_existe = 0;
   end;

   INSERT INTO proclog (proc, fechora) VALUES ('EXISTE_HORA', now());

   SET v_PreHorFchD_ini = STR_TO_DATE(p_desde,'%d/%m/%Y');
   SET v_PreHorFchD_fin = STR_TO_DATE(p_hasta,'%d/%m/%Y');

   if p_uniid = 0 then

      SELECT 1 into p_existe
      FROM horfun
      WHERE cenid = p_cenid AND serid = p_serid AND funid = p_funid AND
```

```
          horfchd >= v_PreHorFchD_ini
        and horfchd < v_PreHorFchD_fin AND horfundisp = 'S'
      LIMIT 1;

  else
      SELECT 1 into p_existe
      FROM horfun
      WHERE cenid = p_cenid AND uniid = p_uniid AND serid = p_serid AND
          funid = p_funid AND horfchd >= v_PreHorFchD_ini
        AND horfchd < v_PreHorFchD_fin and horfundisp = 'S'
      LIMIT 1;
   end if;
end;
//

drop procedure if exists EXISTE_PRESTA;
CREATE PROCEDURE EXISTE_PRESTA(in p_usu_ci VARCHAR(255), in p_Usu_Id
    VARCHAR(255), in p_PreCenId VARCHAR(255), in p_PreHorFchD_ini VARCHAR
    (4000),
    in p_PreHorFchD_fin VARCHAR(4000), in p_PreUniId VARCHAR(255), in
        p_PreSerId VARCHAR(255), out p_existe INT)
begin
   DECLARE v_PreHorFchD_ini VARCHAR(255);
   DECLARE v_PreHorFchD_fin VARCHAR(255);

   DECLARE EXIT HANDLER FOR NOT FOUND
   begin
      SET p_existe = 0;
   end;

   INSERT INTO proclog (proc, fechora) VALUES ('EXISTE_PRESTA', now());

   SET v_PreHorFchD_ini = STR_TO_DATE(p_PreHorFchD_ini,'%d/%m/%Y');
   SET v_PreHorFchD_fin = STR_TO_DATE(p_PreHorFchD_fin,'%d/%m/%Y');

   SELECT 1 into p_existe
   FROM PRESTACION p
   WHERE Usu_CI = p_usu_ci AND Usu_Id = p_Usu_Id AND PreCenId = p_PreCenId
       AND PreHorFchD >= v_PreHorFchD_ini AND PreHorFchD <=
       v_PreHorFchD_fin AND
    PreUniId = p_PreUniId AND PreSerId = p_PreSerId
   LIMIT 1;
end;
//

CREATE PROCEDURE GENERA_ACT()
begin
   DECLARE v_actid VARCHAR(255);
   DECLARE v_actobs VARCHAR(255);
   DECLARE v_ok INT DEFAULT 0;
   DECLARE v_mal INT DEFAULT 0;
   DECLARE v_origen VARCHAR(20) DEFAULT 'ACTPrest1';
   DECLARE v_ci_vieja VARCHAR(15);
   DECLARE v_id_vieja INT;
   DECLARE v_ci_nueva VARCHAR(15);
   DECLARE v_id_nueva INT;

   DECLARE NO_DATA INT DEFAULT 0;
```

```
   DECLARE C_REC cursor FOR
   select CI_VIEJA,ID_VIEJA,CI_NUEVA,ID_NUEVA
   from usu_a_modificar
   where procesado = 0;

   DECLARE CONTINUE HANDLER FOR NOT FOUND SET NO_DATA = -1;

   OPEN C_REC;
   SET NO_DATA = 0;
   FETCH C_REC INTO v_ci_vieja, v_id_vieja, v_ci_nueva, v_id_nueva;

   WHILE NO_DATA = 0 DO
      begin
         DECLARE EXIT HANDLER FOR SQLEXCEPTION
            begin
                SET v_mal = v_mal+1;
            end;

         CALL cambioci(v_ci_vieja, v_id_vieja, v_ci_nueva, v_id_nueva);
         CALL numero(v_origen,v_actid);

         SET v_actobs = CONCAT('Cambia CI ',trim(v_ci_vieja),trim(
             v_id_vieja),' por: ',Trim(v_ci_nueva),trim(v_id_nueva));

         INSERT INTO actuacion(actid,objwid,preid,actfchini,traid,actobs,
             actusuci,actusuid,actusr)
            values(v_actid,1,0,CURRENT_TIMESTAMP,1040, v_actobs,v_ci_vieja,
                v_id_vieja, 'SISTEMA');

         SET v_ok = v_ok+1;

         update usu_a_modificar
         set procesado = 1
         WHERE CI_VIEJA= v_ci_vieja AND ID_VIEJA= v_id_vieja AND CI_NUEVA=
             v_ci_nueva AND ID_NUEVA=v_id_nueva;

      end;
      SET NO_DATA = 0;
      FETCH C_REC INTO v_ci_vieja, v_id_vieja, v_ci_nueva, v_id_nueva;
   END WHILE;
   SET NO_DATA = 0;
   CLOSE C_REC;
   COMMIT;

end;
//

CREATE PROCEDURE GUARDOCAMAS()
begin
   DECLARE EXIT HANDLER FOR NOT FOUND,SQLEXCEPTION
   begin
      rollback;
   end;

   insert into hismov
   select a.cenid,a.uniid,movid,CURRENT_TIMESTAMP,movestid,uniserid
   from movil a, unidad b
   where movtpoid = 4 and b.uniid = a.uniid and b.cenid = a.cenid;
   COMMIT;
```

```
end;
//

drop procedure if exists LEOSTFAREMPA;
CREATE PROCEDURE LEOSTFAREMPA(in V_idfarmacia VARCHAR(255), INOUT v_codigo
    VARCHAR(255), INOUT v_descrip VARCHAR(255),INOUT v_empaque VARCHAR(255)
    ,
    INOUT v_unidad VARCHAR(255),INOUT v_concentrac VARCHAR(255), INOUT
        v_forma_farm VARCHAR(255), INOUT v_cantemp VARCHAR(255))
begin
   DECLARE auxcodigo VARCHAR(255);
   DECLARE auxdescrip VARCHAR(255);
   DECLARE auxemp VARCHAR(255);
   DECLARE auxunidad VARCHAR(255);
   DECLARE unichica CHAR(20);
   DECLARE auxconcentrac VARCHAR(255);
   DECLARE auxforma_farm VARCHAR(255);
   DECLARE auxcantemp VARCHAR(255);
   DECLARE claveuni VARCHAR(255);

   DECLARE aux_idfarmacia VARCHAR(255);

   INSERT INTO proclog (proc, fechora) VALUES ('LEOSTFAREMPA', now());

   begin
      DECLARE EXIT HANDLER FOR NOT FOUND
      begin
         SET auxcodigo = ' ';
         SET auxdescrip = ' ';
         SET auxemp = 0;
         SET auxconcentrac = ' ';
         SET auxforma_farm = 0;
         SET auxcantemp = 0;
         SET auxunidad = ' ';
      end;

      SELECT codigo,descrip,empaque,concentrac,forma_farm,cantemp,unidad
      INTO auxcodigo,auxdescrip,auxemp,auxconcentrac,auxforma_farm,
          auxcantemp,auxunidad
      FROM stfar
      WHERE idfarmacia = V_idfarmacia;
   end;


   SET unichica  = SUBSTRING(auxunidad,1,20);
   SET unichica = trim(unichica);


   if trim(auxunidad) <> ' ' then
      BEGIN
         DECLARE EXIT HANDLER FOR NOT FOUND
         begin
            SET claveuni = 0;
         end;
         select clave into claveuni
         from wf_unidad
         where trim(codigo) = trim(unichica);
```

```
      END;
   end if;
   SET v_codigo  = auxcodigo;
   SET v_descrip = auxdescrip;
   SET v_empaque = auxemp;
   SET v_unidad  = claveuni;
   SET v_concentrac = auxconcentrac;
   SET v_forma_farm = auxforma_farm;
   SET v_cantemp  = auxcantemp;

end;
//

drop procedure if exists REPPER;
CREATE PROCEDURE REPPER ( INOUT pcenid int(11), INOUT puniid int(11), INOUT
     pmovid int(11),
    INOUT pSerId int(11), INOUT phorfchddes datetime, INOUT phorfchdhas
        datetime,
    INOUT prepfch DATETIME, prepperint INT, INOUT prepfunid int(11), INOUT
        pboton VARCHAR(4000),
    INOUT phorfunusrpre VARCHAR(30))

begin
   DECLARE vrepfchtrad DATETIME;
   DECLARE vrepfchtrah DATETIME;
   DECLARE vrepfchhas DATETIME;
   DECLARE vexihorfun VARCHAR(1);
   DECLARE canthorfun MEDIUMINT;
   DECLARE vrepperint SMALLINT;


   DECLARE vhorfundisp VARCHAR(1) DEFAULT 'N';
   DECLARE vhorfunfchpre DATETIME DEFAULT null;
   DECLARE vhorfunusrpre VARCHAR(30) DEFAULT null;
   DECLARE vhorfunfchaus DATETIME DEFAULT null;
   DECLARE vhorfunusraus VARCHAR(30) DEFAULT null;
   DECLARE vserideme VARCHAR(255) DEFAULT 401;

   DECLARE cantferiado TINYINT;


   DECLARE ppcenid;
   DECLARE ppuniid;
   DECLARE ppmovid;
   DECLARE ppserid;
   DECLARE pphorfchddes;
   DECLARE pphorfchdhas;
   DECLARE ppfunid;
   DECLARE NO_DATA INT DEFAULT 0;

   DECLARE SWV_CHORFUN_cenid VARCHAR(255);
   DECLARE SWV_CHORFUN_uniid VARCHAR(255);
   DECLARE SWV_CHORFUN_movid VARCHAR(255);
   DECLARE SWV_CHORFUN_serid VARCHAR(255);
   DECLARE SWV_CHORFUN_horfchd VARCHAR(255);
   DECLARE SWV_CHORFUN_funid VARCHAR(255);
   DECLARE SWV_CHORFUN_rolid VARCHAR(255);
   DECLARE SWV_CHORFUN_horfundisp VARCHAR(255);
   DECLARE SWV_CHORFUN_horfunusrpre VARCHAR(255);
```

```
DECLARE SWV_CHORFUN_horfunfchpre VARCHAR(255);
DECLARE SWV_CHORFUN_horfunusraus VARCHAR(255);
DECLARE SWV_CHORFUN_horfunfchaus VARCHAR(255);
DECLARE SWV_CHORFUN_cauid VARCHAR(255);
DECLARE SWV_CHORA_cenid VARCHAR(255);
DECLARE SWV_CHORA_uniid VARCHAR(255);
DECLARE SWV_CHORA_movid VARCHAR(255);
DECLARE SWV_CHORA_serid VARCHAR(255);
DECLARE SWV_CHORA_horfchd VARCHAR(255);
DECLARE SWV_CHORA_horfchh VARCHAR(255);
DECLARE SWV_CHORA_horcntasig VARCHAR(255);
DECLARE SWV_CHORA_horcntuti VARCHAR(255);
DECLARE SWV_CHORA_hornro VARCHAR(255);
DECLARE SWV_CHORA_hortpocon VARCHAR(255);

DECLARE CHORA CURSOR FOR SELECT h.cenid,h.uniid,h.movid,h.serid,h.
    horfchd,h.horfchh,h.horcntasig,h.horcntuti,h.hornro,h.hortpocon
FROM   hora h
WHERE  CenId = ppcenid and UniId = ppuniid and MovId = ppmovid and SerId
     = ppserid
and  HorFchD >= pphorfchddes and HorFchD < pphorfchdhas;

DECLARE CHORFUN CURSOR FOR SELECT f.cenid,f.uniid,f.movid,f.serid,f.
    horfchd,f.funid,f.rolid,f.horfundisp,f.horfunusrpre,f.horfunfchpre,
     f.horfunusraus,f.horfunfchaus,f.cauid
FROM   horfun f
WHERE  CenId = ppcenid and UniId = ppuniid and MovId = ppmovid and SerId
     = ppserid and HorFchD >= pphorfchddes and
     HorFchD < pphorfchdhas and (Funid = ppfunid or ppfunid = 0);

DECLARE CONTINUE HANDLER FOR NOT FOUND SET NO_DATA = -1;

CALL p_audit_repper(pcenid,puniid,pmovid,pSerId,phorfchddes,phorfchdhas,
    prepfch,prepperint,prepfunid,pboton,phorfunusrpre);

if pSerId <> vserideme then

   SET vhorfundisp   = 'S';
   SET vhorfunusrpre = phorfunusrpre;
   SET vhorfunusraus = phorfunusrpre;
end if;

if pboton = 'P' then

   SET vrepfchhas  = TIMESTAMPADD(day,1,prepfch);
   SET vrepperint  = prepperint;
else
if pboton = 'M' then

      SET vrepfchhas  = TIMESTAMPADD(day,28,prepfch);
      SET vrepperint  = TIMESTAMPDIFF(day,phorfchddes,prepfch);
else
      SET vrepfchhas  = TIMESTAMPADD(day,7,prepfch);
      SET vrepperint  = TIMESTAMPDIFF(day,phorfchddes,prepfch);
   end if;
end if;

SET vexihorfun = 'N';
SET ppcenid = pcenid;
```

```
  SET ppuniid = puniid;
  SET ppmovid = pmovid;
  SET ppserid = pSerId;
  SET pphorfchddes = phorfchddes;
  SET pphorfchdhas = phorfchdhas;
  SET ppfunid = prepfunid;

OPEN CHORFUN;
  SET NO_DATA = 0;
  FETCH CHORFUN INTO SWV_CHORFUN_cenid,SWV_CHORFUN_uniid,SWV_CHORFUN_movid
      ,SWV_CHORFUN_serid,
  SWV_CHORFUN_horfchd,SWV_CHORFUN_funid,SWV_CHORFUN_rolid,
      SWV_CHORFUN_horfundisp,
  SWV_CHORFUN_horfunusrpre,SWV_CHORFUN_horfunfchpre,
      SWV_CHORFUN_horfunusraus,
  SWV_CHORFUN_horfunfchaus,SWV_CHORFUN_cauid;

WHILE NO_DATA = 0 DO
     SET vexihorfun = 'S';

     SET vrepfchtrad = SWV_CHORFUN_horfchd+vrepperint;

 IF pSerId <> vserideme THEN

        SELECT horfchh into vrepfchtrah FROM hora h
        WHERE CenId = SWV_CHORFUN_cenid and UniId = SWV_CHORFUN_uniid and
            MovId = SWV_CHORFUN_movid and
         SerId = SWV_CHORFUN_serid and HorFchD = SWV_CHORFUN_horfchd;
        SET vrepfchtrah = TIMESTAMPADD(day,vrepperint,vrepfchtrah);
     END IF;

 WHILE vrepfchtrad < vrepfchhas DO
        IF pSerId <> 6 and pSerId <> 401 THEN
           SELECT count(*) into cantferiado
           FROM feriados
           WHERE  FERIFECFER_FECHA_FERIADO = STR_TO_DATE(DATE_FORMAT(
               vrepfchtrad,'%d/%m/%Y'),'%d/%m/%Y');
      ELSE
           SET cantferiado = 0;
        END IF;
        IF cantferiado = 0 THEN
           IF pSerId <> vserideme THEN

              SET vhorfunfchpre = vrepfchtrad;
              SET vhorfunfchaus = vrepfchtrah;
           END IF;

           begin
              DECLARE EXIT HANDLER FOR 1062
              begin
                 SET @SWV_Null_Var = 0;
              end;
    INSERT INTO horfun(cenid, uniid, movid, serid, horfchd,funid, rolid,
        horfundisp, horfunusrpre,
     horfunfchpre, horfunusraus, horfunfchaus, cauid)
    VALUES (SWV_CHORFUN_cenid, SWV_CHORFUN_uniid, SWV_CHORFUN_movid,
        SWV_CHORFUN_serid, vrepfchtrad,
     SWV_CHORFUN_funid, SWV_CHORFUN_rolid, vhorfundisp, vhorfunusrpre,
         vhorfunfchpre, vhorfunusraus,
```

```
    vhorfunfchaus, 0);
          end;

       END IF;

       SET vrepfchtrad = TIMESTAMPADD(day,vrepperint,vrepfchtrad);

       IF pSerId <> vserideme THEN
          SET vrepfchtrah = TIMESTAMPADD(day,vrepperint,vrepfchtrah);
       END IF;
    END WHILE;

SET NO_DATA = 0;

FETCH CHORFUN INTO SWV_CHORFUN_cenid,SWV_CHORFUN_uniid,SWV_CHORFUN_movid,
    SWV_CHORFUN_serid,
    SWV_CHORFUN_horfchd,SWV_CHORFUN_funid,SWV_CHORFUN_rolid,
        SWV_CHORFUN_horfundisp,
    SWV_CHORFUN_horfunusrpre,SWV_CHORFUN_horfunfchpre,
        SWV_CHORFUN_horfunusraus,
    SWV_CHORFUN_horfunfchaus,SWV_CHORFUN_cauid;

 END WHILE;

 SET NO_DATA = 0;

 CLOSE CHORFUN;

 IF vexihorfun = 'S' THEN
    SET ppcenid = pcenid;
    SET ppuniid = puniid;
    SET ppmovid = pmovid;
    SET ppserid = pSerId;
    SET pphorfchddes = phorfchddes;
    SET pphorfchdhas = phorfchdhas;

OPEN CHORA;

SET NO_DATA = 0;

FETCH CHORA INTO SWV_CHORA_cenid,SWV_CHORA_uniid,SWV_CHORA_movid,
    SWV_CHORA_serid,SWV_CHORA_horfchd,
    SWV_CHORA_horfchh,SWV_CHORA_horcntasig,SWV_CHORA_horcntuti,
    SWV_CHORA_hornro,SWV_CHORA_hortpocon;

WHILE NO_DATA = 0 DO
       SET vrepfchtrad = SWV_CHORA_horfchd+vrepperint;
       SET vrepfchtrah = SWV_CHORA_horfchh+vrepperint;
       while vrepfchtrad < vrepfchhas DO
          if pSerId <> 6 and pSerId <> 401 then
             select count(*) into cantferiado from feriados
             where  FERIFECFER_FECHA_FERIADO = STR_TO_DATE(DATE_FORMAT(
                 vrepfchtrad,'%d/%m/%Y'),'%d/%m/%Y');
       else
             SET cantferiado = 0;
          end if;
          if cantferiado = 0 then
             SET canthorfun = 0;
```

```
                begin
                   DECLARE EXIT HANDLER FOR NOT FOUND
                   begin
                      SET @SWV_Null_Var = 0;
                   end;
                   DECLARE EXIT HANDLER FOR 1328
                   begin
                      DECLARE EXIT HANDLER FOR 1062
                      begin
                         SET @SWV_Null_Var = 0;
                      end;
insert into hora(cenid,uniid,movid,serid,horfchd,horfchh,horcntasig,
    horcntuti,hornro,hortpocon)
values(SWV_CHORA_cenid,SWV_CHORA_uniid,SWV_CHORA_movid,SWV_CHORA_serid,
    vrepfchtrad,vrepfchtrah,
SWV_CHORA_horcntasig,0,SWV_CHORA_hornro,SWV_CHORA_hortpocon);
                   end;
                   select 1 into canthorfun from horfun
                   where CenId = SWV_CHORA_cenid and UniId = SWV_CHORA_uniid
                        and MovId = SWV_CHORA_movid and
                   SerId = SWV_CHORA_serid and HorFchD = vrepfchtrad;



                end;
             end if;
             SET vrepfchtrad = TIMESTAMPADD(day,vrepperint,vrepfchtrad);
             SET vrepfchtrah = TIMESTAMPADD(day,vrepperint,vrepfchtrah);
          END WHILE;
          SET NO_DATA = 0;
          FETCH CHORA INTO SWV_CHORA_cenid,SWV_CHORA_uniid,SWV_CHORA_movid,
              SWV_CHORA_serid,SWV_CHORA_horfchd,
          SWV_CHORA_horfchh,SWV_CHORA_horcntasig,SWV_CHORA_horcntuti,
          SWV_CHORA_hornro,SWV_CHORA_hortpocon;
       END WHILE;
       SET NO_DATA = 0;
       CLOSE CHORA;
    end if;
    COMMIT;

end;
//

CREATE PROCEDURE MUEVECONSULTAS()
begin

    DECLARE fchMin VARCHAR(10);
    DECLARE fchMax VARCHAR(10);

    DECLARE EXIT HANDLER FOR NOT FOUND,SQLEXCEPTION
    begin
       rollback;
    end;

    SET fchMin = DATE_FORMAT(TIMESTAMPADD(day,-10,CURRENT_TIMESTAMP),'%d/%m
        /%Y');
    SET fchMax = DATE_FORMAT(TIMESTAMPADD(day,-9,CURRENT_TIMESTAMP),'%d/%m/%
```

```
    Y');

savepoint prueba;

update prestacion set
preestid = 1003
where preestid = 1002
and prehorfchd >= STR_TO_DATE(fchMin,'%d/%m/%Y')
and prehorfchd < STR_TO_DATE(fchMax,'%d/%m/%Y')
and (prediagdsc is null or trim(prediagdsc) = '')
and preserid <> 503
and preserid <> 604
and preserid <> 504
and preserid <> 602
and preserid <> 650;

COMMIT;

update prestacion set
preestid = 1007
where preestid = 1002
and prehorfchd >= STR_TO_DATE(fchMin,'%d/%m/%Y')
and prehorfchd < STR_TO_DATE(fchMax,'%d/%m/%Y')
and not prediagdsc is null
and preserid <> 503
and preserid <> 604
and preserid <> 504
and preserid <> 602
and preserid <> 650;

COMMIT;


update prestacion set
preestid = 1003
where preestid = 1002
and prehorfchd >= STR_TO_DATE(fchMin,'%d/%m/%Y')
and prehorfchd < STR_TO_DATE(fchMax,'%d/%m/%Y')
and (rcr_hiscli is null or trim(rcr_hiscli) = '')
and (preserid = 503 or preserid = 604 or preserid = 504);

COMMIT;

update prestacion set
preestid = 1007
where preestid = 1002
and prehorfchd >= STR_TO_DATE(fchMin,'%d/%m/%Y')
and prehorfchd < STR_TO_DATE(fchMax,'%d/%m/%Y')
and not rcr_hiscli is null
and (preserid = 503 or preserid = 604 or preserid = 504);

COMMIT;


update prestacion set
preestid = 1003
where preestid = 1002
and prehorfchd >= STR_TO_DATE(fchMin,'%d/%m/%Y')
and prehorfchd < STR_TO_DATE(fchMax,'%d/%m/%Y')
```

```
   and not exists(select b.preid from preproc b where b.preid = prestacion.
       preid)
   and preserid = 602;

   COMMIT;

   update prestacion set
   preestid = 1007
   where preestid = 1002
   and prehorfchd >= STR_TO_DATE(fchMin,'%d/%m/%Y')
   and prehorfchd < STR_TO_DATE(fchMax,'%d/%m/%Y')
   and exists(select b.preid from preproc b where b.preid = prestacion.
       preid)
   and preserid = 602;

   COMMIT;


   update prestacion set
   preestid = 1003
   where preestid = 1002
   and prehorfchd >= STR_TO_DATE(fchMin,'%d/%m/%Y')
   and prehorfchd < STR_TO_DATE(fchMax,'%d/%m/%Y')
   and (premotdsc is null or trim(premotdsc) = '')
   and preserid = 650;

   COMMIT;

   update prestacion set
   preestid = 1007
   where preestid = 1002
   and prehorfchd >= STR_TO_DATE(fchMin,'%d/%m/%Y')
   and prehorfchd < STR_TO_DATE(fchMax,'%d/%m/%Y')
   and not premotdsc is null
   and preserid = 650;

   COMMIT;

end;
//


CREATE PROCEDURE MUTUALISTAS()
begin
   DECLARE fchMin DATETIME;
   DECLARE fchMax DATETIME;
   DECLARE fchMinDes DATETIME;
   DECLARE fchMaxDes DATETIME;

   SET fchMin = CURDATE();
   SET fchMax = CURDATE()+1;


   SET fchMinDes = CURDATE()-1;
   SET fchMaxDes = CURDATE();


   SET @SWV_Stmt = 'truncate table auxmut';
   PREPARE SWT_Stmt FROM @SWV_Stmt;
```

```
   EXECUTE SWT_Stmt;
   DEALLOCATE PREPARE SWT_Stmt;


   insert into auxmut
   select preid,usu_ci,usu_id
   from prestacion, servicio
   where prehorfchd >= fchMin and prehorfchd < fchMax and preserid = serid
       and serprod > 0 and preserid <> 10003
        and preestid = 1002;

   COMMIT;


   insert into auxmut
   select distinct preid,pin_usuci,pin_usuid
   from preins
   where pin_fecha >= fchMinDes and pin_fecha < fchMaxDes and pin_serid =
       10003 and pin_estid = 2216;

   COMMIT;


   insert into auxmut
   select preid,usu_ci,usu_id
   from prestacion
   where prehorfchd >= fchMinDes and prehorfchd < fchMaxDes and preserid =
       401 and precenid = 900
        and preestid = 5006;

   COMMIT;

   begin
      DECLARE EXIT HANDLER FOR NOT FOUND,SQLEXCEPTION
      begin
         rollback;
      end;

      savepoint premut;


      insert into premut
      select distinct preid,b.mutnro
      from auxmut a, rucaf b, mutual c
      where rucci = CAST(SUBSTRING(usu_ci,1,length(usu_ci) -1) AS unsigned)
           and rucid = CAST(SUBSTRING(usu_ci,length(usu_ci),1) AS unsigned)
          and c.mutnro = b.mutnro and (mutnorucaf = 'S' OR (mutnorucaf = 'N'
               and usu_id = 0));

      COMMIT;

   end;



end;
```

C.3.2. Triggers

```
CREATE TRIGGER CENFUN_JNTRG_I
AFTER INSERT ON cenfun FOR EACH ROW

INSERT INTO cenfun_jn (CENID, UNIID, FUNID, CENFUNFCH, aud_action, aud_date
    , aud_user) values
(new.CENID, new.UNIID, new.FUNID, new.CENFUNFCH, 'INS', NOW() , USER());


CREATE TRIGGER CENFUN_JNTRG_U
AFTER UPDATE ON CENFUN FOR EACH ROW

INSERT INTO cenfun_jn (CENID, UNIID, FUNID, CENFUNFCH, aud_action, au_date,
     aud_user) values
(new.CENID, new.UNIID, new.FUNID, new.CENFUNFCH, 'UPD', NOW() , USER());


CREATE TRIGGER CENFUN_JNTRG_D
AFTER DELETE ON CENFUN FOR EACH ROW

INSERT INTO cenfun_jn (CENID, UNIID, FUNID, CENFUNFCH, aud_action, au_date,
     aud_user) values
(old.CENID, old.UNIID, old.FUNID, old.CENFUNFCH, 'DEL', NOW() , USER());
```

Anexo D

Bug hallado en Genexus en la generación del código para PostgreSQL

La aplicación generada por Genexus para su uso con la base de datos Postgres, presentó un bug que impedía el correcto funcionamiento del caso de uso "Programar oferta de servicios", así como del caso "Atender Solicitud de Consulta".

En ambos casos, se desplegaba la excepción que se muestra en la imagen:

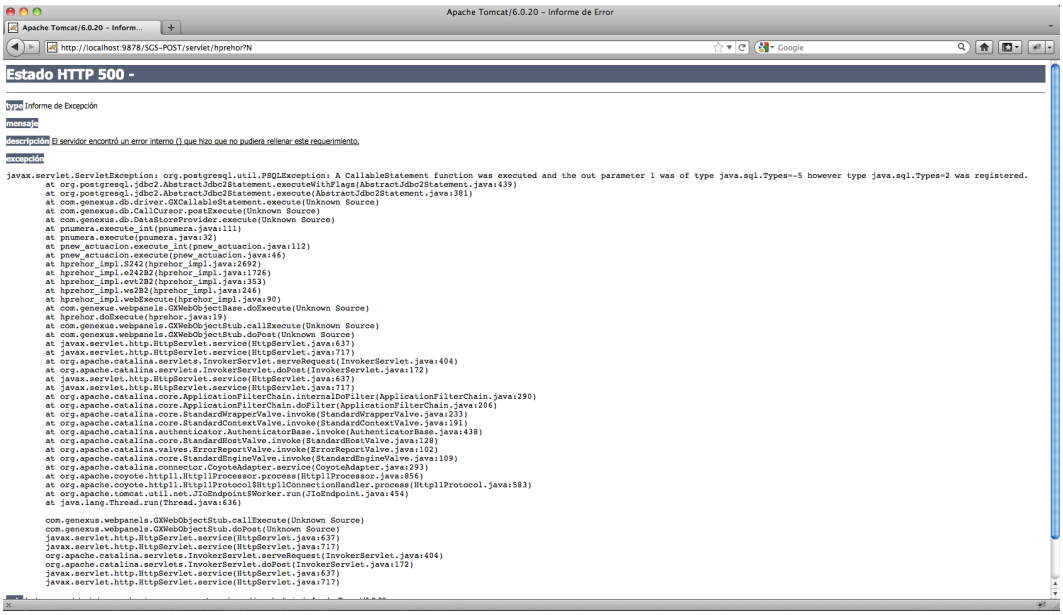

Figura D.1: Excepción java: conflicto de tipos de datos

Los tipos de datos involucrados se corresponden con los tipos *bigint* (tipo -5) y *numeric* (tipo 2).

En primer lugar, para el análisis del código involucrado en la excepción desplegada, se debió decompilar las clases correspondientes. Para ello se utilizó el decompilador JAD de Pavel Kouznetsov. Según surge del análisis de la clase pnumera, la excepción se dispara tras la llamada al *stored procedure* NUMERO en la clase pnumera__default:

```
return (new Cursor[] {
           new ForEachCursor("P00052",
"SELECT NumId, NumOri, NumUlt FROM public.NUMERA WHERE (NumOri = ? AND NumId = ?)
  AND (NumOri = ? and NumId = ?)
 FOR UPDATE OF NUMERA",
true, 16, false, this, 0, true),
new UpdateCursor("P00053",
"SAVEPOINT gxupdate;UPDATE public.NUMERA SET NumUlt=?  WHERE NumOri = ? AND
NumId = ?;RELEASE SAVEPOINT gxupdate", 80),
new UpdateCursor("P00054", "SAVEPOINT gxupdate;
INSERT INTO public.NUMERA (NumOri, NumId, NumDsc, NumUlt) VALUES (?, ?, ?, ?);
RELEASE SAVEPOINT gxupdate", 80),
new CallCursor("P00055", "{? = CALL NUMERO ( ?, ?)}", 16)
```

El código del procedimiento NUMERO es el siguiente:

```
CREATE OR REPLACE FUNCTION numero(INOUT pnumero bigint, IN porigen
    character varying)
  RETURNS bigint AS
$BODY$

BEGIN
if porigen = 'PREPrest1' then
select nextval('num_prestacion') into pnumero from dual;
end if;
if porigen = 'ACTUsuar4' then
select nextval('num_actuacion_usu') into pnumero from dual;
end if;
if porigen = 'ACTMovil5' then
select nextval('num_actuacion_mov') into pnumero from dual;
end if;
if porigen = 'ACTSaldo6' then
select nextval('num_actuacion_sal') into pnumero from dual;
end if;
if porigen = 'ACTPrest1' then
select nextval('num_actuacion_pre') into pnumero from dual;
end if;
END;

$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION numero(bigint, character varying) OWNER TO postgres;
```

El intento de sustituir el tipo del parámetro de salida por *numeric* en lugar del *bigint* existente, también fracasa disparando una nueva excepción según se ve en la figura siguiente:

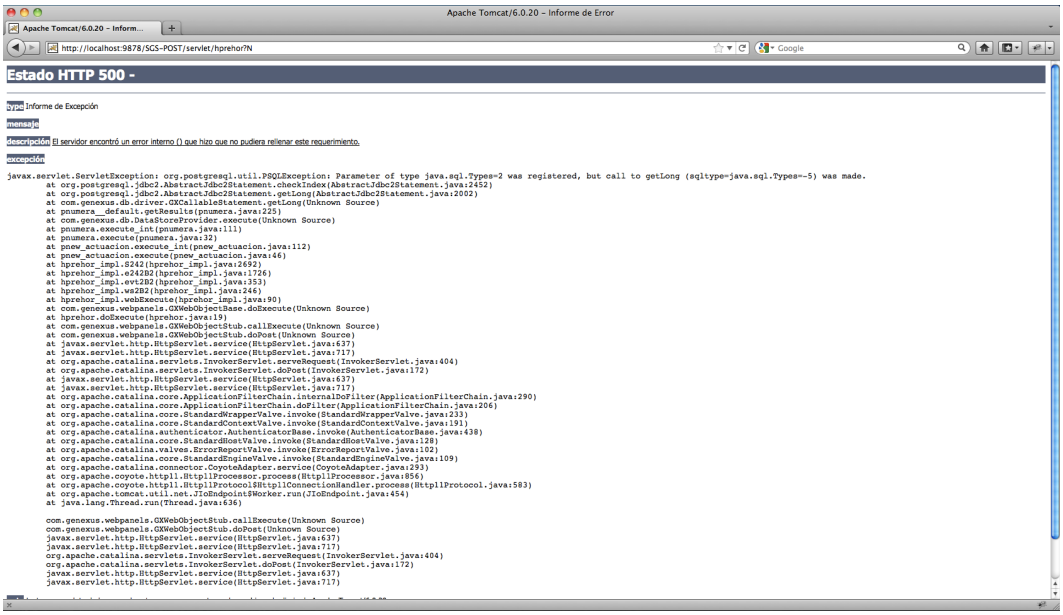

Figura D.2: Excepción java: conflicto de tipos de datos 2

Si bien, la sustitución del tipo de datos por *numeric* resuelve el problema de la llamada al *stored procedure*, el problema se traslada al momento en el que se intenta procesar el dato recibido. La documentación de java señala la equivalencia entre el tipo de datos *bigint* en SQL con el tipo de datos Long en java, de lo que se infiere que el tipo de datos a usar debe ser *bigint* y no *numeric* y el error está en el momento de registrar el tipo de datos, como se veía en la primera figura.

En el llamado al método setParameters de la clase pnumera__default aparece el registro del parámetro en cuestión:

```
public void setParameters(int i, IFieldSetter ifieldsetter, Object aobj
    [])
     throws SQLException
 {
     switch(i)
     {
     default:
         break;

     case 0:
         ifieldsetter.setInt(1, ((Number)aobj[0]).intValue());
         ifieldsetter.setVarchar(2, (String)aobj[1], 10);
         ifieldsetter.setInt(3, ((Number)aobj[2]).intValue());
         ifieldsetter.setVarchar(4, (String)aobj[3], 10);
         break;

     case 1:
         if(((Boolean)aobj[0]).booleanValue())
             ifieldsetter.setNull(1, 2);
         else
             ifieldsetter.setLong(1, ((Number)aobj[1]).longValue());
         ifieldsetter.setInt(2, ((Number)aobj[2]).intValue());
         ifieldsetter.setVarchar(3, (String)aobj[3], 10, false);
```

```
            break;

        case 2:
            ifieldsetter.setInt(1, ((Number)aobj[0]).intValue());
            ifieldsetter.setVarchar(2, (String)aobj[1], 10, false);
            if(((Boolean)aobj[2]).booleanValue())
                ifieldsetter.setNull(3, 12);
            else
                ifieldsetter.setVarchar(3, (String)aobj[3], 30);
            if(((Boolean)aobj[4]).booleanValue())
                ifieldsetter.setNull(4, 2);
            else
                ifieldsetter.setLong(4, ((Number)aobj[5]).longValue());
            break;

        case 3:
            ((GXCallableStatement)ifieldsetter).registerOutParameter(1, 2,
                0);
            ifieldsetter.setLong(2, ((Number)aobj[1]).longValue());
            ifieldsetter.setVarchar(3, (String)aobj[2], 10);
            break;
        }
    }
```

Al sustituir el tipo 2 (*numeric*) del registro del parámetro de salida por el tipo -5 (*bigint*), el problema queda solucionado, pudiéndose completar los dos casos de uso en cuestión.

Anexo E

Emulación de sequences en MySQL

E.1. Tabla

Cada registro de la tabla utilizada para el manejo de las *sequences* presenta un campo para el nombre de la *sequence*, otro para el valor actual utilizado, otros dos para máximo y mínimo valor posible, uno para el incremento y por último un campo que permite establecer si la secuencia será cíclica o no.

```
mysql> explain sequence_data;
+--------------------+---------------------+------+-----+----------------------+-------+
| Field              | Type                | Null | Key | Default              | Extra |
+--------------------+---------------------+------+-----+----------------------+-------+
| sequence_name      | varchar(100)        | NO   | PRI | NULL                 |       |
| sequence_increment | int(11) unsigned    | NO   |     | 1                    |       |
| sequence_min_value | int(11) unsigned    | NO   |     | 1                    |       |
| sequence_max_value | bigint(20) unsigned | NO   |     | 18446744073709551615 |       |
| sequence_cur_value | bigint(20) unsigned | YES  |     | 1                    |       |
| sequence_cycle     | tinyint(1)          | NO   |     | 0                    |       |
+--------------------+---------------------+------+-----+----------------------+-------+
6 rows in set (0.00 sec)
```

E.2. Funciones para emulación de sequences

A continuación se muestra el código utilizado para la función "_seq_nextval" que devuelve el siguiente valor de la secuencia. Nótese que no se ha tenido en cuenta la necesidad de cuidar la transaccionalidad dado que el único uso es permitir el funcionamiento correcto de la aplicación.

```
delimiter //
create function nextval( seq_name varchar(32) )
returns integer unsigned deterministic
begin
        select sequence_max_value into @maxval from sequence_data where
            name=sequence_name;
```

```
        select sequence_cur_value into @curval from sequence_data where
            name=sequence_name;
        select sequence_increment into @step from sequence_data where name=
            sequence_name;

        if @curval + @step > @maxval then
           SET @curval = (@curval + @step) % @maxval;
        else
           SET @curval = @curval + @step;
        end if;
        update sequence_data set sequence_cur_value = @curval;
        return @curval;
end//
```

Las funciones currval(sequence) y lastval(sequence) se implementan con un SELECT sobre el campo correspondiente y la función setval(sequence, value) con el INSERT correspondiente, ateniéndose a las restricciones dadas por los valores "sequence_min_value" y "sequence_max_value".

Anexo F

MonetDB y el intento fallido de integración con el SGS

Si bien existe un driver de MonetDB para Java, Genexus no cuenta con un conector para interactuar con este DBMS. De cualquier forma se intentó conectar las plataformas confiando en la compatibilidad de los conectores de MySQL y PostgreSQL.

ASSE cuenta con suporte por parte de Artech, canalizado a través de Sergio Gutiérrez, integrante del equipo de desarrollo especialista en la herramienta.

A continuación se muestra el hilo de discusión sobre este asunto.

```
from: Martin Prunell martin.prunell@gmail.com
to: sergio.gutierrez@asse.com.uy,
    ariel sabiguero yawelak <ariel.sabiguero@asse.com.uy>,
    Lorena Etcheverry <lorenae@fing.edu.uy>,
    Nelson Calero <nelson.calero@asse.com.uy>
cc: Leonardo Mendez <lmaya03@gmail.com>, alexhg <alexhg@gmail.com>
date: Fri, Nov 19, 2010 at 7:04 PM
subject: MonetDB y GX
mailed-by: gmail.com


Buenas,

Resumiendo lo que hablamos con Sergio sobre MonetDB y GX: en principio
sería difícil realizar la conexión ya que GX no tiene a MonetDB entre las
bases de datos soportadas.

Tenemos las siguientes opciones:

1. Crear el WAR para la BD que sea "más parecida" a MonetDB, y luego
alterar la conexión, a ver qué pasa.

2. Enviar un mail a soporte de GX para preguntar de qué manera se puede
```

conectar a MonetDB, teniendo el driver.

¿Qué les parece? En caso de que ninguna de las opciones nos lleven a ningún lado, lo que es muy probable, por lo menos nos queda el resultado para documentación, no?

---

from: Sergio Gutiérrez sergio.gutierrez@asse.com.uy
to: Martin Prunell <martin.prunell@gmail.com>
cc: ariel sabiguero yawelak <ariel.sabiguero@asse.com.uy>,
    Lorena Etcheverry <lorenae@fing.edu.uy>,
    Nelson Calero <nelson.calero@asse.com.uy>
date: Mon, Nov 22, 2010 at 10:37 AM
subject: Re: MonetDB y GX
mailed-by: asse.com.uy

Martín:

Me parece que se podría probar la opción 1 como primera instancia, según el DBMS Genexus arma las consultas correspondientes entonces, si las consultas que maneja un DBMS y otro son muy parecidas, se puede generar un war solamente cambiando el client.cfg.
Oracle y MySQL maneja las consultas muy parecidas y el 80 % del sistema podría funcionar solamente cambiando ese archivo. Entonces avisen a cual es más parecido y generamos el War.Una vez probado esto si no funciona o funciona muy poco, enviamos el mail a Artech con argumentos sobre lo que hicimos para que nos den un Ok o no definitivo.

Gracias,

Sergio

---

from: Sergio Gutiérrez sergio.gutierrez@asse.com.uy
to: Martin Prunell <martin.prunell@gmail.com>
cc: ariel sabiguero yawelak <ariel.sabiguero@asse.com.uy>,
    Lorena Etcheverry <lorenae@fing.edu.uy>,
    Nelson Calero <nelson.calero@asse.com.uy>
date: Fri, Feb 11, 2011 at 3:12 PM
subject: Re: MonetDB y GX
mailed-by: asse.com.uy

```
Buenas:

Como comentario el jdbc de monet para conectarme es:
jdbc:monetdb://sgs-foss:50000/sgs-test.
Luego el classname es: nl.cwi.monetdb.jdbc.MonetDriver
Les paso los errores que me dió, uno es usando el WAR de MySQL y otro de
Postgres. Por lo que veo con MySQL ni siquiera pudo encontrar el driver
como que ya lo necesitaba
predefinido y con Postgress me parece que que ya encontró el driver y
esta ejecutando un query y hay error en el begin dentro del stored
procedure. Le paso a Ariel de todas formas Monet con generación para
Postgres. Cualquier cosa avisen.

Saludos,
```

F.1. Pruebas con conectores de MySQL y PostgreSQL

F.1.1. MySQL

Los intentos de conexión del SGS a MonetDB fueron infructuosos, los errores se muestran a continuación. Al fallar este "workaround" no se encontró la forma de realizar la integración utilizando los medios disponibles dentro del alcance de este trabajo.

```
MySQL

Estado HTTP 500 -

type Informe de Excepción
mensaje
descripción El servidor encontró un error interno () que hizo que no
pudiera rellenar este requerimiento.
excepción
javax.servlet.ServletException: java.lang.NullPointerException
at com.genexus.db.driver.GXDBMSmysql.DataTruncation
(Unknown Source)
at com.genexus.db.DefaultExceptionErrorHandler.mapErrorToStatus
(Unknown Source)
at com.genexus.db.DefaultExceptionErrorHandler.handleSQLError
(Unknown Source)
at com.genexus.db.DataStoreProvider.execute(Unknown Source)
at pdetpuesto.execute_int(pdetpuesto.java:56)
at pdetpuesto.execute(pdetpuesto.java:33)
at hmenu_impl.e13332(hmenu_impl.java:838)
at hmenu_impl.rf332(hmenu_impl.java:630)
at hmenu_impl.refresh(hmenu_impl.java:623)
at hmenu_impl.we332(hmenu_impl.java:577)
at hmenu_impl.webExecute(hmenu_impl.java:66)
```

```
at com.genexus.webpanels.GXWebObjectBase.doExecute
(Unknown Source)
at hmenu.doExecute(hmenu.java:29)
at com.genexus.webpanels.GXWebObjectStub.callExecute
(Unknown Source)
at com.genexus.webpanels.GXWebObjectStub.doGet(Unknown Source)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:627)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:729)
at org.apache.catalina.servlets.InvokerServlet.serveRequest
(InvokerServlet.java:420)
at org.apache.catalina.servlets.InvokerServlet.doGet
(InvokerServlet.java:134)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:627)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:729)
at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter
(ApplicationFilterChain.java:269)
at org.apache.catalina.core.ApplicationFilterChain.doFilter
(ApplicationFilterChain.java:188)
at org.apache.catalina.core.StandardWrapperValve.invoke
(StandardWrapperValve.java:213)
at org.apache.catalina.core.StandardContextValve.invoke
(StandardContextValve.java:172)
at org.apache.catalina.authenticator.AuthenticatorBase.invoke
(AuthenticatorBase.java:558)
at org.apache.catalina.core.StandardHostValve.invoke
(StandardHostValve.java:127)
at org.apache.catalina.valves.ErrorReportValve.invoke
(ErrorReportValve.java:117)
at org.apache.catalina.core.StandardEngineValve.invoke
(StandardEngineValve.java:108)
at org.apache.catalina.connector.CoyoteAdapter.service
(CoyoteAdapter.java:174)
at org.apache.coyote.http11.Http11Processor.process
(Http11Processor.java:873)
at org.apache.coyote.http11.Http11BaseProtocol$
Http11ConnectionHandler.processConnection(Http11BaseProtocol.java:665)
at org.apache.tomcat.util.net.PoolTcpEndpoint.processSocket
(PoolTcpEndpoint.java:528)
at org.apache.tomcat.util.net.LeaderFollowerWorkerThread.runIt
(LeaderFollowerWorkerThread.java:81)
at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run
(ThreadPool.java:689)
at java.lang.Thread.run(Unknown Source)

com.genexus.webpanels.GXWebObjectStub.callExecute(Unknown Source)
com.genexus.webpanels.GXWebObjectStub.doGet(Unknown Source)
javax.servlet.http.HttpServlet.service(HttpServlet.java:627)
```

```
javax.servlet.http.HttpServlet.service(HttpServlet.java:729)
org.apache.catalina.servlets.InvokerServlet.serveRequest
(InvokerServlet.java:420)
org.apache.catalina.servlets.InvokerServlet.doGet
(InvokerServlet.java:134)
javax.servlet.http.HttpServlet.service(HttpServlet.java:627)
javax.servlet.http.HttpServlet.service(HttpServlet.java:729)
nota La traza completa de la causa de este error se encuentra en los
archivos de diario de Apache Tomcat/5.5.29.

Apache Tomcat/5.5.29
```

F.1.2. PostgreSQL

```
Estado HTTP 500 -

type Informe de Excepción
mensaje
descripción El servidor encontró un error interno () que hizo que no
pudiera rellenar este requerimiento.
excepción
javax.servlet.ServletException: java.sql.SQLException: syntax error,
unexpected BEGIN in: "begin"
at nl.cwi.monetdb.jdbc.MonetConnection$ResponseList.executeQuery
(MonetConnection.java:2208)
at nl.cwi.monetdb.jdbc.MonetConnection$ResponseList.processQuery
(MonetConnection.java:1967)
at nl.cwi.monetdb.jdbc.MonetStatement.internalExecute
(MonetStatement.java:434)
at nl.cwi.monetdb.jdbc.MonetStatement.execute
(MonetStatement.java:347)
at nl.cwi.monetdb.jdbc.MonetStatement.executeUpdate
(MonetStatement.java:470)
at com.genexus.db.driver.GXStatement.executeUpdate
(Unknown Source)
at com.genexus.db.driver.GXDBMSpostgresql.onConnection
(Unknown Source)
at com.genexus.db.driver.GXConnection.<init>(Unknown Source)
at com.genexus.db.driver.GXConnection.<init>(Unknown Source)
at com.genexus.db.driver.ReadWriteConnectionPool.createConnection
(Unknown Source)
at com.genexus.db.driver.ConnectionPool.checkOut(Unknown Source)
at com.genexus.db.driver.DataSourceConnectionPool.checkOut
(Unknown Source)
at com.genexus.db.ServerUserInformation.getConnection
```

```
(Unknown Source)
at com.genexus.db.ServerDBConnectionManager.getConnection
(Unknown Source)
at com.genexus.db.DefaultConnectionProvider.getConnection
(Unknown Source)
at com.genexus.db.DataStoreProviderBase.getConnection
(Unknown Source)
at com.genexus.db.SentenceProvider.getPreparedStatement
(Unknown Source)
at com.genexus.db.ForEachCursor.preExecute(Unknown Source)
at com.genexus.db.DataStoreProvider.execute(Unknown Source)
at pdetpuesto.execute_int(pdetpuesto.java:56)
at pdetpuesto.execute(pdetpuesto.java:33)
at hmenu_impl.e13332(hmenu_impl.java:836)
at hmenu_impl.rf332(hmenu_impl.java:628)
at hmenu_impl.refresh(hmenu_impl.java:621)
at hmenu_impl.we332(hmenu_impl.java:575)
at hmenu_impl.webExecute(hmenu_impl.java:66)
at com.genexus.webpanels.GXWebObjectBase.doExecute
(Unknown Source)
at hmenu.doExecute(hmenu.java:29)
at com.genexus.webpanels.GXWebObjectStub.callExecute
(Unknown Source)
at com.genexus.webpanels.GXWebObjectStub.doGet(Unknown Source)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:627)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:729)
at org.apache.catalina.servlets.InvokerServlet.serveRequest
(InvokerServlet.java:420)
at org.apache.catalina.servlets.InvokerServlet.doGet
(InvokerServlet.java:134)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:627)
at javax.servlet.http.HttpServlet.service(HttpServlet.java:729)
at org.apache.catalina.core.ApplicationFilterChain.
internalDoFilter(ApplicationFilterChain.java:269)
at org.apache.catalina.core.ApplicationFilterChain.
doFilter(ApplicationFilterChain.java:188)
at org.apache.catalina.core.StandardWrapperValve.invoke
(StandardWrapperValve.java:213)
at org.apache.catalina.core.StandardContextValve.invoke
(StandardContextValve.java:172)
at org.apache.catalina.authenticator.AuthenticatorBase.invoke
(AuthenticatorBase.java:558)
at org.apache.catalina.core.StandardHostValve.invoke
(StandardHostValve.java:127)
at org.apache.catalina.valves.ErrorReportValve.invoke
(ErrorReportValve.java:117)
at org.apache.catalina.core.StandardEngineValve.invoke
```

```
(StandardEngineValve.java:108)
at org.apache.catalina.connector.CoyoteAdapter.service
(CoyoteAdapter.java:174)
at org.apache.coyote.http11.Http11Processor.process
(Http11Processor.java:873)
at org.apache.coyote.http11.Http11BaseProtocol$
Http11ConnectionHandler.processConnection(Http11BaseProtocol.java:665)
at org.apache.tomcat.util.net.PoolTcpEndpoint.processSocket
(PoolTcpEndpoint.java:528)
at org.apache.tomcat.util.net.LeaderFollowerWorkerThread.runIt
(LeaderFollowerWorkerThread.java:81)
at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.
run(ThreadPool.java:689)
at java.lang.Thread.run(Unknown Source)

com.genexus.webpanels.GXWebObjectStub.callExecute(Unknown Source)
com.genexus.webpanels.GXWebObjectStub.doGet(Unknown Source)
javax.servlet.http.HttpServlet.service(HttpServlet.java:627)
javax.servlet.http.HttpServlet.service(HttpServlet.java:729)
org.apache.catalina.servlets.InvokerServlet.serveRequest
(InvokerServlet.java:420)
org.apache.catalina.servlets.InvokerServlet.doGet
(InvokerServlet.java:134)
javax.servlet.http.HttpServlet.service(HttpServlet.java:627)
javax.servlet.http.HttpServlet.service(HttpServlet.java:729)
nota La traza completa de la causa de este error se encuentra en
los archivos de diario de Apache Tomcat/5.5.29.

Apache Tomcat/5.5.29
```

Bibliografía

[1] Mysql architecture cluster. http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster.html, Fecha de última comprobación: 24/06/2012.

[2] Partitioning tables in mysql. http://dev.mysql.com/doc/refman/5.1/en/partitioning.html, Fecha de última comprobación: 24/06/2012.

[3] Postgresql: Hotbackup. http://pghotbackup.projects.postgresql.org/, Fecha de última comprobación: 24/06/2012.

[4] MySQL Data Base. Page official. WebSite. http://www.mysql.com, Fecha de última comprobación: 24/06/2012.

[5] Josh Berkus. Why postrgresql doesn't have query hints. Technical report, PostgreSQL, 2011. http://it.toolbox.com/blogs/database-soup/why-postgresql-doesnt-have-query-hints-44121.

[6] ELMASARI/NAVATHE. *Sistemas de Base de Datos*. Addison Wesley, segunda edition, 2005.

[7] Ewen Fortune. Beware of running analyze in production, 2011. http://www.mysqlperformanceblog.com/2011/02/03/how-innodb-handles-redo-logging/, Fecha de última comprobación: 24/06/2012.

[8] Peter Gulutzan. Mysql 5.0 storedprocedures. Technical report, MySQL AB, Marzo 2005.

[9] Ispirer Systems Ltd. SQLWAYS - Migrating Oracle to MySQL. http://www.ispirer.com/products/oracle-to-mysql-migration, Fecha de última comprobación: 24/06/2012.

[10] MySQL 5.0 Reference Manual. SHOW INNODB STATUS y los monitores InnoDB. WebSite. http://dev.mysql.com/doc/refman/5.0/es/innodb-monitor.html, Fecha de última comprobación: 24/06/2012.

[11] MySQL 5.1 Reference Manual. Chapter 14. Storage Engines. WebSite. http://dev.mysql.com/doc/refman/5.1/en/storage-engines.html, Fecha de última comprobación: 24/06/2012.

[12] MySQL 5.1 Reference Manual. Innodb Storage Engine. WebSite. http://dev.mysql.com/doc/refman/5.1/en/innodb-storage-engine.html, Fecha de última comprobación: 24/06/2012.

[13] MySQL 5.1 Reference Manual. Myisam Storage Engine. WebSite. http://dev.mysql.com/doc/refman/5.1/en/myisam-storage-engine.html, Fecha de última comprobación: 24/06/2012.

[14] MySQL 5.1. Reference Manual. Optimize table syntax. http://dev.mysql.com/doc/refman/5.1/en/optimize-table.html, Fecha de última comprobación: 24/06/2012.

[15] MySQL 5.1. Reference Manual. Server System Variables. http://dev.mysql.com/doc/refman/5.1/en/server-system-variables.html, Fecha de última comprobación: 24/06/2012.

[16] MonetDB. The Column Store Pioneer. WebSite. http://www.monetdb.org/Home, Fecha de última comprobación: 24/06/2012.

[17] Chuck Murray. *Oracle Database SQL Developer Supplementary Information for MySQL Migrations, Release 1.5*, 2008.

[18] MySQL. Modelo de bloqueo transaccional en mysql. http://mirror.neu.edu.cn/mysql/doc/refman/5.1/en/innodb-transaction-model.html, Fecha de última comprobación: 24/06/2012.

[19] MySQL. Mysql 5.0 reference manual. http://dev.mysql.com/doc/refman/5.0/en/innodb-tuning.html, Fecha de última comprobación: 24/06/2012.

[20] MySQL. Page official. http://www.mysql.com/products/cluster/, Fecha de última comprobación: 24/06/2012.

[21] Oracle. Oracle database application developer's guide - fundamentals. http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14251/adfns_triggers.htm, Fecha de última comprobación: 24/06/2012.

[22] Oracle. Oracle GoldenGate. http://www.oracle.com/us/products/middleware/data-integration/goldengate/overview/index.html, Fecha de última comprobación: 24/06/2012.

[23] Percona. Percona - xtrabackup. Manual Reference. http://http://www.percona.com/doc/percona-xtrabackup/, Fecha de última comprobación: 24/06/2012.

[24] PostgreSQL. Page official. http://www.postgresql.org, Fecha de última comprobación: 24/06/2012.

[25] PostgreSQL. Page official. http://www.postgresql.org/about/users/, Fecha de última comprobación: 24/06/2012.

[26] PostgreSQL. Postgresql: Control de concurrencia. http://www.postgresql.org/docs/8.4/interactive/mvcc.html, Fecha de última comprobación: 24/06/2012.

[27] PostgreSQL. Postgresql documentation - chapter 5.9. Manual Reference. http://www.postgresql.org/docs/8.4/static/ddl-partitioning.html, Fecha de última comprobación: 24/06/2012.

[28] PostgreSQL. Postgresql: Documentation: Manuals: Porting from oracle pl/sql. http://www.postgresql.org/docs/8.3/interactive/plpgsql-porting.html, Fecha de última comprobación: 24/06/2012.

[29] PostgreSQL. 14.3. controlling the planner with explicit join clauses. Technical report, PostgreSQL, 2010.

[30] PostgreSQL. Documentation–manuals–postgresql 8.3–continuous archiving and point-in-time recovery (pitr). Technical report, PostgreSQL, 2010.

[31] PostrgreSQL. Page official. http://www.postgresql.org/docs/8.2/static/server-shutdown.html, Fecha de última comprobación: 24/06/2012.

[32] P. Griffiths Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access path selection in a relational database system. Technical report, IBM Research Division, 1979.

[33] Greg Smith, Robert Treat, and Christopher Browne. Tuning your postgresql server. http://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server, Fecha de última comprobación: 24/06/2012.

[34] SquirrelSQL. Squirrelsql. http://squirrel-sql.sourceforge.net/, Fecha de última comprobación: 24/06/2012.

[35] Michael Stonebracker. The design of the postgres storage system. Technical report, University of California.

[36] Peng Sun, Huang Jian, Markus Wanner, and Tomoaki Sato. Pgfoundry: pgpool: Información del proyecto. http://pgfoundry.org/projects/pgpool/, Fecha de última comprobación: 24/06/2012.

[37] Heikki Tuuri. Concurrency control: How it really works, 2009. http://www.innodb.com/wp/wp-content/uploads/2009/05/concurrencycontrol.pdf, Fecha de última comprobación: 24/06/2012.

[38] Vertica. Columnar store. http://www.vertica.com/2010/04/22/column-store-vs-column-store/, Fecha de última comprobación: 24/06/2012.

[39] Xaprb. Better cacti templates. WebSite. http://code.google.com/p/mysql-cacti-templates/, Fecha de última comprobación: 24/06/2012.

[40] Peter Zaitsev. Why mysql could be slow with large tables?, 2006. http://www.mysqlperformanceblog.com/2006/06/09/why-mysql-could-be-slow-with-large-tables/, Fecha de última comprobación: 24/06/2012.

[41] Peter Zaitsev. Beware of running analyze in production, 2008. http://www.mysqlperformanceblog.com/2008/09/02/beware-of-running-analyze-in-production/, Fecha de última comprobación: 24/06/2012.

[42] Peter Zaitsev. Beware of running analyze in productionthinking about running optimize on your innodb table ? stop!, 2010. http://www.mysqlperformanceblog.com/2010/12/09/thinking-about-running-optimize-on-your-innodb-table-stop/, Fecha de última comprobación: 24/06/2012.

Glosario

Glossary

ACID *Atomicity, Consistency, Isolation, Durability*: conjunto de propiedades que garantizan que una transacción en una base de datos sea procesada en forma confiable. Atomicidad: requiere que una transacción sea "todo o nada". Si una parte de la transacción falla, la transacción falla completamente, y la base de datos se mantiene incambiada. Consistencia: propiedad que asegura que una una transacción pondrá a la base de datos en un estado válido, según sus reglas definidas. Aislamiento: propiedad que asegura que dos transacciones no interferirán entre sí. Si operan sobre el mismo conjunto de datos, una deberá esperar por que la otra termine. Durabilidad: propiedad que asegura el estado de los datos. Una vez que la transacción fue confirmada, el estado de los datos permanecerá incambiado hasta que ocurra una nueva transacción que lo altere.. 7, 42

ANSI/ISO SQL *ANSI/ISO SQL*, El ANSI adoptó SQL como estándar para los lenguajes relacionales y en 1987 se transformó en estándar ISO. Esta versión del estándar va con el nombre de SQL/86. En los años siguientes, éste ha sufrido diversas revisiones que han conducido primero a la versión SQL/89 y, posteriormente, a la actual SQL/92.. 42

ASSE *Administración de los Servicios de Salud del Estado*: principal prestador de servicios integrales de salud estatales en la República Oriental del Uruguay.. 1, 68, 71

BAT *Binary Association Tables* Estructura de almacenamiento utilizada por MonetDB en el que cada columna de una tabla es almacenada por separado y accedida por intermedio de esta estructura.. 12

DBA *Database Administrator*: profesional responsable por la instalación, configuración, actualización, administración, monitoreo y mantenimiento de una base de datos.. 4, 68

FOSS *Free Open Source Software*: Software que con licenciamiento libre y de código abierto.. 3, 7

Genexus Herramienta de desarrollo orientada a aplicaciones enterprise para Web, Microsoft Windows y dispositivos inteligentes. El desarrollador describe la lógica del negocio utilizando un lenguaje declarativo que es traducido a lenguaje nativo por Genexus.. 1, 7, 57, 72

GNU GPL *General Public Licence*: es una licencia creada por la Free Software Foundation y está orientada principalmente a proteger la libre distribución, modificación y uso del software.. 8

Internet Explorer Navegador (Browser) web gráfico desarrollado para ejecutarse en los sistemas operativos Microsoft Windows.. 1

ISAM *Indexed Sequential Access Method*: Método para almacenar información de forma tal que se pueda acceder rápidamente a ella.. 8

Java Lenguaje de programación basado en la sintaxis de C/C++ que corre sobre una máquina virtual, siendo de esta forma independiente de la arquitectura del computador.. 1, 25

Knowledge Base *Knowledge Base*: Repositorio de información abierto o cerrado utilizado en Genexus para almacenar la especificación de los objetos. 57

MIL *MonetDB Interpreter Language* Lenguage de bajo nivel interpretado por el kernel de MonetDB que actúa de interfaz entre sus operaciones nativas e interfaces externas, siendo una de estas SQL.. 12

MVCC *Multi-Version Concurrency Control* Técnica utilizada evitar problemas de concurrencia y consistencia de datos cuando dos o mas transacciones acceden al mismo recurso.. 43

ODBC *Open Data Base Connectivity*: es un estándar de acceso a bases de datos independiente del motor utilizado. 26

OLAP *Online Analytical Processing*: es un enfoque para devolver resultados de consultas multidimensionales analíticas sobre datos multidimensionales.. 11

OLTP *Online Transaction Processing*: sistemas orientados a transacciones.. 8

Oracle Sistema Manejador de Base de Datos relacionales privativo desarrollado por Oracle Corporation.. 1

Percona Percona es una consultora de MySQL independiente que provee soporte y servicios de ingeniería.. 32

RDBMS *Relational Database Management System*: Sistema Manejador de Base de Datos del tipo relacional.. 3, 7, 29, 57

RRDTool *Round Robin Database Tool*: Herramienta que recolecta datos periodicamente sobre estado de la red y CPU y otros. Cada uno de los datos es elegido siguiendo el algoritmo Round Robin (cola circular) de manera que la imagen del estado del sistema se mantenga constante a lo largo del tiempo.. 19

SGS Sistema de Gestión de Salud, es una aplicación que permite la gestión de emergencias, policlínicas y farmacias, así como la creación de agendas para los médicos y la asignación de números a los pacientes.. 7

SNMP *Simple Network Management Protocol*: (Protocolo Simple de Adminsitración de Red): protocolo de red de capa de aplicación que facilita el intercambio de información de administración entre despositivas de red.. 19

Tomcat (también Apache Tomcat): Contenedor de aplicaciones Java.. 1

WAL *Write-Ahead Log* enfoque estándard de registro de transacciones de una base de datos. Radica en que cualquier cambio en los archivos de datos puede ser escritos únicamente después de que haya sido registrado de forma permanente.. 24

Índice de figuras

Índice de cuadros